Ying Tan
Yuhui Shi
Yi Chai
Guoyin Wang (Eds.)

LNCS 6728

Advances in Swarm Intelligence

Second International Conference, ICSI 2011
Chongqing, China, June 2011
Proceedings, Part I



Springer

Lecture Notes in Computer Science 6728

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Alfred Kobsa
University of California, Irvine, CA, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
TU Dortmund University, Germany

Madhu Sudan
Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Ying Tan Yuhui Shi Yi Chai
Guoyin Wang (Eds.)

Advances in Swarm Intelligence

Second International Conference, ICSI 2011
Chongqing, China, June 12-15, 2011
Proceedings, Part I

Springer

Volume Editors

Ying Tan
Peking University
Key Laboratory of Machine Perception (MOE)
Department of Machine Intelligence
Beijing, 100871, China
E-mail: ytan@pku.edu.cn

Yuhui Shi
Xi'an Jiaotong-Liverpool University
Department of Electrical and Electronic Engineering
Suzhou, 215123, China
E-mail: yuhui.shi@xjtlu.edu.cn

Yi Chai
Chongqing University
Automation College
Chongqing 400030, China
E-mail: chaiyi@cqu.edu.cn

Guoyin Wang
Chongqing University of Posts and Telecommunications
College of Computer Science and Technology
Chongqing, 400065, China
E-mail: wanggy@ieee.org

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-21514-8 e-ISBN 978-3-642-21515-5
DOI 10.1007/978-3-642-21515-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011928465

CR Subject Classification (1998): F.1, H.3, I.2, H.4, H.2.8, I.4-5

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

© Springer-Verlag Berlin Heidelberg 2011
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book and its companion volume, LNCS vols. 6728 and 6729, constitute the proceedings of the Second International Conference on Swarm Intelligence (ICSI 2011) held during June 12–15, 2011 in Chongqing, well known as the Mountain City, the southwestern commercial capital of China. ICSI 2011 was the second gathering in the world for researchers working on all aspects of swarm intelligence, following the successful and fruitful Beijing ICSI event in 2010, which provided a high-level international academic forum for the participants to disseminate their new research findings and discuss emerging areas of research. It also created a stimulating environment for the participants to interact and exchange information on future challenges and opportunities in the field of swarm intelligence research.

ICSI 2011 received 298 submissions from about 602 authors in 38 countries and regions (Algeria, American Samoa, Argentina, Australia, Austria, Belize, Bhutan, Brazil, Canada, Chile, China, Germany, Hong Kong, Hungary, India, Islamic Republic of Iran, Japan, Republic of Korea, Kuwait, Macau, Madagascar, Malaysia, Mexico, New Zealand, Pakistan, Romania, Saudi Arabia, Singapore, South Africa, Spain, Sweden, Chinese Taiwan, Thailand, Tunisia, Ukraine, UK, USA, Vietnam) across six continents (Asia, Europe, North America, South America, Africa, and Oceania). Each submission was reviewed by at least 2 reviewers, and on average 2.8 reviewers. Based on rigorous reviews by the Program Committee members and reviewers, 143 high-quality papers were selected for publication in the proceedings with an acceptance rate of 47.9%. The papers are organized in 23 cohesive sections covering all major topics of swarm intelligence research and development.

In addition to the contributed papers, the ICSI 2011 technical program included four plenary speeches by Russell C. Eberhart (Indiana University Purdue University Indianapolis (IUPUI), USA), K. C. Tan (National University of Singapore, Singapore, the Editor-in-Chief of IEEE Computational Intelligence Magazine (CIM)), Juan Luis Fernandez Martnez (University of Oviedo, Spain), Fernando Buarque (University of Pernambuco, Brazil). Besides the regular oral sessions, ICSI 2011 had two special sessions on 'Data Fusion and Swarm Intelligence' and 'Fish School Search Foundations and Application' as well as several poster sessions focusing on wide areas.

As organizers of ICSI 2011, we would like to express sincere thanks to Chongqing University, Peking University, Chongqing University of Posts and Telecommunications, and Xi'an Jiaotong-Liverpool University for their sponsorship, to the IEEE Computational Intelligence Society, World Federation on Soft Computing, International Neural Network Society, and Chinese Association for Artificial Intelligence for their technical co-sponsorship. We appreciate the Natural Science Foundation of China for its financial and logistic supports.

We would also like to thank the members of the Advisory Committee for their guidance, the members of the International Program Committee and additional reviewers for reviewing the papers, and members of the Publications Committee for checking the accepted papers in a short period of time. Particularly, we are grateful to the proceedings publisher Springer for publishing the proceedings in the prestigious series of *Lecture Notes in Computer Science*. Moreover, we wish to express our heartfelt appreciation to the plenary speakers, session chairs, and student helpers. There are still many more colleagues, associates, friends, and supporters who helped us in immeasurable ways; we express our sincere gratitude to them all. Last but not the least, we would like to thank all the speakers and authors and participants for their great contributions that made ICSI 2011 successful and all the hard work worthwhile.

June 2011

Ying Tan
Yuhui Shi
Yi Chai
Guoyin Wang

Organization

General Chairs

Russell C. Eberhart	Indiana University - Purdue University, USA
Dan Yang	Chongqing University, China
Ying Tan	Peking University, China

Advisory Committee Chairs

Xingui He	Peking University, China
Qidi Wu	Tongji University, China
Gary G. Yen	Oklahoma State University, USA

Program Committee Chairs

Yuhui Shi	Xi'an Jiaotong-Liverpool University, China
Guoyin Wang	Chongqing University of Posts and Telecommunications, China

Technical Committee Chairs

Yi Chai	Chongqing University, China
Andries Engelbrecht	University of Pretoria, South Africa
Nikola Kasabov	Auckland University of Technology, New Zealand
Kay Chen Tan	National University of Singapore, Singapore
Peng-yeng Yin	National Chi Nan University, Taiwan, China
Martin Middendorf	University of Leipzig, Germany

Plenary Sessions Chairs

Xiaohui Cui	Oak Ridge National Laboratory, USA
James Tin-Yau Kwok	The Hong Kong University of Science and Technology, China

Special Sessions Chairs

Majid Ahmadi	University of Windsor, Canada
Hongwei Mo	Harbin Engineering University, China
Yi Zhang	Sichuan University, China

Publications Chairs

Rajkumar Roy	Cranfield University, UK
Radu-Emil Precup	Politehnica University of Timisoara, Romania
Yue Sun	Chongqing University, China

Publicity Chairs

Xiaodong Li	RMIT Unversity, Australia
Haibo He	University of Rhode Island Kingston, USA
Lei Wang	Tongji University, China
Weiren Shi	Chongqing University, China
Jin Wang	Chongqing University of Posts and Telecommunications, China

Finance Chairs

Chao Deng	Peking University, China
Andreas Janecek	University of Vienna, Austria

Local Arrangements Chairs

Dihua Sun	Chongqing University, China
Qun Liu	Chongqing University of Posts and Telecommunications, China

Program Committee Members

Payman Arabshahi	University of Washington, USA
Carmelo Bastos	University of Pernambuco, Brazil
Christian Blum	Universitat Politecnica de Catalunya, Spain
Leandro Leandro dos Santos Coelho	Pontifícia Universidade Católica do Parana, Brazil
Carlos Coello Coello	CINVESTAV-IPN, Mexico
Oscar Cordon	European Centre for Soft Computing, Spain
Jose Alfredo Ferreira Costa	UFRN Universidade Federal do Rio Grande do Norte, Brazil
Iain Couzin	Princeton University, USA
Xiaohui Cui	Oak Ridge National Laboratory, USA
Swagatam Das	Jadavpur University, India
Prithviraj Dasgupta	University of Nebraska, USA
Kusum Deep	Indian Institute of Technology Roorkee, India
Mingcong Deng	Okayama University, Japan
Haibin Duan	Beijing University of Aeronautics and Astronautics, China

Mark Embrechts	RPI, USA
Andries Engelbrecht	University of Pretoria, South Africa
Wai-Keung Fung	University of Manitoba, Canada
Beatriz Aurora Garro Licon	CIC-IPN, Mexico
Dunwei Gong	China University of Mining and Technology, China
Ping Guo	Beijing Normal University, China
Walter Gutjahr	University of Vienna, Austria
Qing-Long Han	Central Queensland University, Australia
Haibo He	University of Rhode Island, USA
Lu Hongtao	Shanghai Jiao Tong University, China
Mo Hongwei	Harbin Engineering University, China
Zeng-Guang Hou	Institute of Automation, Chinese Academy of Sciences, China
Huosheng Hu	University of Essex, UK
Guang-Bin Huang	Nanyang Technological University, Singapore
Yuancheng Huang	Wuhan University, China
Hisao Ishibuchi	Osaka Prefecture University, Japan
Andreas Janecek	University of Vienna, Austria
Zhen Ji	Shenzhen University, China
Changan Jiang	Kagawa University, Japan
Licheng Jiao	Xidian University, China
Colin Johnson	University of Kent, UK
Farrukh Aslam Khan	FAST-National University of Computer and Emerging Sciences, Pakistan
Arun Khosla	National Institute of Tech. Jalandhar, India
Franziska Klügl	Örebro University, Sweden
James Kwok	Hong Kong University of Science and Technology, China
Xiaodong Li	RMIT University, Australia
Yangmin Li	University of Macau, China
Fernando Buarque De Lima Neto	Polytechnic School of Pernambuco, Brazil
Guoping Liu	University of Glamorgan, UK
Ju Liu	Shandong University, China
Qun Liu	Chongqing University of Posts and Communications, China
Wenlian Lu	Fudan University, China
Juan Luis Fernandez Martinez	University of Oviedo, Spain
Wenjian Luo	University of Science and Technology of China, China
Jinwen Ma	Peking University, China
Bernd Meyer	Monash University, Australia

Martin Middendorf	University of Leipzig, Germany
Mahamed G. H. Omran	Gulf University for Science and Technology, Kuwait
Jeng-Shyang Pan Pan	National Kaohsiung University of Applied Sciences, Taiwan, China
Shaoning Pang	Auckland University of Technology, New Zealand
Bijaya Ketan Panigrahi	IIT Delhi, India
Thomas Potok	ORNL, USA
Radu-Emil Precup	Politehnica University of Timisoara, Romania
Guenter Rudolph	TU Dortmund University, Germany
Gerald Schaefer	Loughborough University, UK
Yuhui Shi	Xi'an Jiaotong-Liverpool University, China
Michael Small	Hong Kong Polytechnic University, China
Jim Smith	University of the West of England, UK
Ponnuthurai Suganthan	Nanyang Technological University, Singapore
Norikazu Takahashi	Kyushu University, Japan
Kay-Chen Tan	National University of Singapore, Singapore
Ying Tan	Peking University, China
Ke Tang	University of Science and Technology of China, China
Peter Tino	University of Birmingham, UK
Christos Tjortjis	The University of Manchester, UK
Frans Van Den Bergh	CSIR, South Africa
Ba-Ngu Vo	The University of Western Australia, Australia
Bing Wang	University of Hull, UK
Guoyin Wang	Chongqing University of Posts and Telecommunications, China
Hongbo Wang	Yanshan University, China
Jiahai Wang	Sun Yat-sen University, China
Jin Wang	Chongqing University of Posts and Telecommunications, China
Lei Wang	Tongji University, China
Ling Wang	Tsinghua University, China
Lipo Wang	Nanyang Technological University, Singapore
Benlian Xu	Changshu Institute of Technology, China
Pingkun Yan	Philips Research North America, USA
Yingjie Yang	De Montfort University, UK
Hoengpeng Yin	Chongqing University, China
Peng-Yeng Yin	National Chi Nan University, Taiwan, China
Dingli Yu	Liverpool John Moores University, UK
Jie Zhang	Newcastle University, UK
Jun Zhang	Waseda University, Japan
Lifeng Zhang	Renmin University of China, China
Qieshi Zhang	Waseda University, Japan
Qingfu Zhang	University of Essex, UK

Dongbin Zhao	Institute of Automation, Chinese Academy of Science, China
Zhi-Hua Zhou	Nanjing University, China

Additional Reviewers

Bi, Chongke
Cheng, Chi Tai
Damas, Sergio
Ding, Ke
Dong, Yongsheng
Duong, Tung
Fang, Chonglun
Guo, Jun
Henmi, Tomohiro
Hu, Zhaohui
Huang, Sheng-Jun
Kalra, Gaurav
Lam, Franklin
Lau, Meng Cheng
Leung, Carson K.
Lu, Qiang
Nakamura, Yukinori
Osunleke, Ajiboye
Qing, Li
Quirin, Arnaud
Saleem, Muhammad
Samad, Rosdiyana
Sambo, Francesco
Singh, Satvir
Sun, Fuming
Sun, Yang
Tang, Yong
Tong, Can
Vázquez, Roberto A.
Wang, Hongyan
Wang, Lin
Yanou, Akira
Zhang, Dawei
Zhang, X.M.
Zhang, Yong
Zhu, Yanqiao

Table of Contents – Part I

Theoretical Analysis of Swarm Intelligence Algorithms

Particle Swarm Optimization: A Powerful Family of Stochastic Optimizers. Analysis, Design and Application to Inverse Modelling 1
Juan Luis Fernández-Martínez, Esperanza García-Gonzalo, Saras Saraswathi, Robert Jernigan, and Andrzej Kloczkowski

Building Computational Models of Swarms from Simulated Positional Data 9
Graciano Dieck Kattas and Michael Small

Robustness and Stagnation of a Swarm in a Cooperative Object Recognition Task 19
David King and Philip Breedon

Enforced Mutation to Enhancing the Capability of Particle Swarm Optimization Algorithms 28
PenChen Chou and JenLian Chen

Normalized Population Diversity in Particle Swarm Optimization 38
Shi Cheng and Yuhui Shi

Particle Swarm Optimization with Disagreements 46
Andrei Lihu and Ştefan Holban

PSOslope: A Stand-Alone Windows Application for Graphical Analysis of Slope Stability 56
Walter Chen and Powen Chen

A Review of the Application of Swarm Intelligence Algorithms to 2D Cutting and Packing Problem 64
Yanxin Xu, Gen Ke Yang, Jie Bai, and Changchun Pan

Particle Swarm Optimization

Inertia Weight Adaption in Particle Swarm Optimization Algorithm 71
Zheng Zhou and Yuhui Shi

Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization 80
Wudai Liao, Junyan Wang, and Jiangfeng Wang

An Adaptive Tribe-Particle Swarm Optimization 86
Yong Duan Song, Lu Zhang, and Peng Han

A Novel Hybrid Binary PSO Algorithm 93
Muhammd Ilyas Menhas, MinRui Fei, Ling Wang, and Xiping Fu

PSO Algorithm with Chaos and Gene Density Mutation for Solving Nonlinear Zero-One Integer Programming Problems 101
Yuelin Gao, Fanfan Lei, Huirong Li, and Jimin Li

A New Binary PSO with Velocity Control 111
Laura Lanzarini, Javier López, Juan Andrés Maulini, and Armando De Giusti

Adaptive Particle Swarm Optimization Algorithm for Dynamic Environments 120
Iman Rezazadeh, Mohammad Reza Meybodi, and Ahmad Naebi

An Improved Particle Swarm Optimization with an Adaptive Updating Mechanism 130
Jie Qi and Yongsheng Ding

Mortal Particles: Particle Swarm Optimization with Life Span 138
Yong-wei Zhang, Lei Wang, and Qi-di Wu

Applications of PSO Algorithms

PSO Based Pseudo Dynamic Method for Automated Test Case Generation Using Interpreter 147
Surender Singh Dahiya, Jitender Kumar Chhabra, and Shakti Kumar

Reactive Power Optimization Based on Particle Swarm Optimization Algorithm in 10kV Distribution Network 157
Chao Wang, Gang Yao, Xin Wang, Yihui Zheng, Lidan Zhou, Qingshan Xu, and Xinyuan Liang

Clustering-Based Particle Swarm Optimization for Electrical Impedance Imaging 165
Gang Hu, Min-you Chen, Wei He, and Jin-qian Zhai

A PSO- Based Robust Optimization Approach for Supply Chain Collaboration with Demand Uncertain 172
Yutian Jia, Xingquan Zuo, and Jianping Wu

A Multi-valued Discrete Particle Swarm Optimization for the Evacuation Vehicle Routing Problem 182
Marina Yusoff, Junaidah Ariffin, and Azlinah Mohamed

A NichePSO Algorithm Based Method for Process Window Selection ... 194
Wenqi Li, Yiming Qiu, Lei Wang, and Qidi Wu

Efficient WiFi-Based Indoor Localization Using Particle Swarm Optimization ... 203
Girma S. Tewolde and Jaerock Kwon

Using PSO Algorithm for Simple LSB Substitution Based Steganography Scheme in DCT Transformation Domain ... 212
Feno Heriniaina Rabevohitra and Jun Sang

Numerical Integration Method Based on Particle Swarm Optimization ... 221
Leila Djerou, Naceur Khelil, and Mohamed Batouche

Identification of VSD System Parameters with Particle Swarm Optimization Method ... 227
Yiming Qiu, Wenqi Li, Dongsheng Yang, Lei Wang, and Qidi Wu

PSO-Based Emergency Evacuation Simulation ... 234
Jialiang Kou, Shengwu Xiong, Hongbing Liu, Xinlu Zong, Shuzhen Wan, Yi Liu, Hui Li, and Pengfei Duan

Training Spiking Neurons by Means of Particle Swarm Optimization ... 242
Roberto A. Vázquez and Beatriz A. Garro

Ant Colony Optimization Algorithms

Clustering Aggregation for Improving Ant Based Clustering ... 250
Akil Elkamel, Mariem Gzara, and Hanêne Ben-Abdallah

Multi-cellular-ant Algorithm for Large Scale Capacity Vehicle Route Problem ... 260
Jie Li, Yi Chai, Penghua Li, and Hongpeng Yin

Ant Colony Optimization for Global White Matter Fiber Tracking ... 267
Yuanjing Feng and Zhejin Wang

Bee Colony Algorithms

An Efficient Bee Behavior-Based Multi-function Routing Algorithm for Network-on-Chip ... 277
Junhui Wang, Huaxi Gu, Yintang Yang, and Zhi Deng

Artificial Bee Colony Based Mapping for Application Specific Network-on-Chip Design ... 285
Zhi Deng, Huaxi Gu, Haizhou Feng, and Baojian Shu

Using Artificial Bee Colony to Solve Stochastic Resource Constrained Project Scheduling Problem 293
Amin Tahooneh and Koorush Ziarati

Novel Swarm-Based Optimization Algorithms

Brain Storm Optimization Algorithm 303
Yuhui Shi

Human Group Optimizer with Local Search 310
Chaohua Dai, Weirong Chen, Lili Ran, Yi Zhang, and Yu Du

Average-Inertia Weighted Cat Swarm Optimization 321
Maysam Orouskhani, Mohammad Mansouri, and Mohammad Teshnehlab

Standby Redundancy Optimization with Type-2 Fuzzy Lifetimes 329
Yanju Chen and Ying Liu

Oriented Search Algorithm for Function Optimization 338
Xuexia Zhang and Weirong Chen

Evolution of Cooperation under Social Norms in Non-Structured Populations 347
Qi Xiaowei, Ren Guang, Yue Gin, and Zhang Aiping

Collaborative Optimization under a Control Framework for ATSP 355
Jie Bai, Jun Zhu, Gen-Ke Yang, and Chang-Chun Pan

Bio-Inspired Dynamic Composition and Reconfiguration of Service-Oriented Internetware Systems 364
Huan Zhou, Zili Zhang, Yuheng Wu, and Tao Qian

A Novel Search Interval Forecasting Optimization Algorithm 374
Yang Lou, Junli Li, Yuhui Shi, and Linpeng Jin

Artificial Immune System

A Danger Theory Inspired Learning Model and Its Application to Spam Detection 382
Yuanchun Zhu and Ying Tan

Research of Hybrid Biogeography Based Optimization and Clonal Selection Algorithm for Numerical Optimization 390
Zheng Qu and Hongwei Mo

The Hybrid Algorithm of Biogeography Based Optimization and Clone Selection for Sensors Selection of Aircraft 400
Lifang Xu, Shouda Jiang, and Hongwei Mo

A Modified Artificial Immune Network for Feature Extracting 408
Hong Ge and XueMing Yan

Differential Evolution

Novel Binary Encoding Differential Evolution Algorithm 416
Changshou Deng, Bingyan Zhao, Yanling Yang, Hu Peng, and Qiming Wei

Adaptive Learning Differential Evolution for Numeric Optimization 424
Yi Liu, Shengwu Xiong, Hui Li, and Shuzhen Wan

Differential Evolution with Improved Mutation Strategy 431
Shuzhen Wan, Shengwu Xiong, Jialiang Kou, and Yi Liu

Gaussian Particle Swarm Optimization with Differential Evolution Mutation 439
Chunqiu Wan, Jun Wang, Geng Yang, and Xing Zhang

Neural Networks

Evolving Neural Networks: A Comparison between Differential Evolution and Particle Swarm Optimization 447
Beatriz A. Garro, Humberto Sossa, and Roberto A. Vázquez

Identification of Hindmarsh-Rose Neuron Networks Using GEO metaheuristic 455
Lihe Wang, Genke Yang, and Lam Fat Yeung

Delay-Dependent Stability Criterion for Neural Networks of Neutral-Type with Interval Time-Varying Delays and Nonlinear Perturbations 464
Guoquan Liu, Simon X. Yang, and Wei Fu

Application of Generalized Chebyshev Neural Network in Air Quality Prediction 472
Fengjun Li

Financial Time Series Forecast Using Neural Network Ensembles 480
Anupam Tarsauliya, Rahul Kala, Ritu Tiwari, and Anupam Shukla

Selection of Software Reliability Model Based on BP Neural Network ... 489
Yingbo Wu and Xu Wang

Genetic Algorithms

Atavistic Strategy for Genetic Algorithm 497
Dongmei Lin, Xiaodong Li, and Dong Wang

An Improved Co-evolution Genetic Algorithm for Combinatorial Optimization Problems 506
Nan Li and Yi Luo

Recursive Structure Element Decomposition Using Migration Fitness Scaling Genetic Algorithm 514
Yudong Zhang and Lenan Wu

A Shadow Price Guided Genetic Algorithm for Energy Aware Task Scheduling on Cloud Computers 522
Gang Shen and Yan-Qing Zhang

A Solution to Bipartite Drawing Problem Using Genetic Algorithm 530
Salabat Khan, Mohsin Bilal, Muhammad Sharif, and Farrukh Aslam Khan

Evolutionary Computation

Evaluation of Two-Stage Ensemble Evolutionary Algorithm for Numerical Optimization 539
Yu Wang, Bin Li, Kaibo Zhang, and Zhen He

A Novel Genetic Programming Algorithm For Designing Morphological Image Analysis Method 549
Jun Wang and Ying Tan

Fuzzy Methods

Optimizing Single-Source Capacitated FLP in Fuzzy Decision Systems 559
Liwei Zhang, Yankui Liu, and Xiaoqing Wang

New Results on a Fuzzy Granular Space 568
Xu-Qing Tang and Kun Zhang

Fuzzy Integral Based Data Fusion for Protein Function Prediction 578
Yinan Lu, Yan Zhao, Xiaoni Liu, and Yong Quan

Hybrid Algorithms

Gene Clustering Using Particle Swarm Optimizer Based Memetic Algorithm 587
Zhen Ji, Wenmin Liu, and Zexuan Zhu

Hybrid Particle Swarm Optimization with Biased Mutation Applied to Load Flow Computation in Electrical Power Systems 595
Camila Paes Salomon, Maurilio Pereira Coutinho, Germano Lambert-Torres, and Cláudio Ferreira

Simulation of Routing in Nano-Manipulation for Creating Pattern with Atomic Force Microscopy Using Hybrid GA and PSO-AS Algorithms ... 606
Ahmad Naebi, Moharam Habibnejad Korayem, Farhoud Hoseinpour, Sureswaran Ramadass, and Mojtaba Hoseinzadeh

Neural Fuzzy Forecasting of the China Yuan to US Dollar Exchange Rate—A Swarm Intelligence Approach ... 616
Chunshien Li, Chuan Wei Lin, and Hongming Huang

A Hybrid Model for Credit Evaluation Problem ... 626
Hui Fu and Xiaoyong Liu

Author Index ... 635

Table of Contents – Part II

Multi-Objective Optimization Algorithms

Multi-Objective Optimization for Dynamic Single-Machine Scheduling 1
Li Nie, Liang Gao, Peigen Li, and Xiaojuan Wang

Research of Pareto-Based Multi-Objective Optimization for Multi-vehicle Assignment Problem Based on MOPSO 10
Ai Di-Ming, Zhang Zhe, Zhang Rui, and Pan Feng

Correlative Particle Swarm Optimization for Multi-objective Problems 17
Yuanxia Shen, Guoyin Wang, and Qun Liu

A PSO-Based Hybrid Multi-Objective Algorithm for Multi-Objective Optimization Problems 26
Xianpeng Wang and Lixin Tang

The Properties of Birandom Multiobjective Programming Problems 34
Yongguo Zhang, Yayi Xu, Mingfa Zheng, and Liu Ningning

A Modified Multi-objective Binary Particle Swarm Optimization Algorithm 41
Ling Wang, Wei Ye, Xiping Fu, and Muhammad Ilyas Menhas

Improved Multiobjective Particle Swarm Optimization for Environmental/Economic Dispatch Problem in Power System 49
Yali Wu, Liqing Xu, and Jingqian Xue

A New Multi-Objective Particle Swarm Optimization Algorithm for Strategic Planning of Equipment Maintenance 57
Haifeng Ling, Yujun Zheng, Ziqiu Zhang, and Xianzhong Zhou

Multiobjective Optimization for Nurse Scheduling 66
Peng-Yeng Yin, Chih-Chiang Chao, and Ya-Tzu Chiang

A Multi-objective Binary Harmony Search Algorithm 74
Ling Wang, Yunfei Mao, Qun Niu, and Minrui Fei

Multi-robot, Swarm-robot, and Multi-agent Systems

A Self-organized Approach to Collaborative Handling of Multi-robot Systems 82
Tian-yun Huang, Xue-bo Chen, Wang-bao Xu, and Wei Wang

An Enhanced Formation of Multi-robot Based on A* Algorithm for Data Relay Transmission 91
Zhiguang Xu, Kyung-Sik Choi, Yoon-Gu Kim, Jinung An, and Suk-Gyu Lee

WPAN Communication Distance Expansion Method Based on Multi-robot Cooperation Navigation 99
Yoon-Gu Kim, Jinung An, Kyoung-Dong Kim, Zhi-Guang Xu, and Suk-Gyu Lee

Relative State Modeling Based Distributed Receding Horizon Formation Control of Multiple Robot Systems 108
Wang Zheng, He Yuqing, and Han Jianda

Simulation and Experiments of the Simultaneous Self-assembly for Modular Swarm Robots 118
Hongxing Wei, Yizhou Huang, Haiyuan Li, and Jindong Tan

Impulsive Consensus in Networks of Multi-agent Systems with Any Communication Delays 128
Quanjun Wu, Li Xu, Hua Zhang, and Jin Zhou

Data Mining Methods

FDClust: A New Bio-inspired Divisive Clustering Algorithm 136
Besma Khereddine and Mariem Gzara

Mining Class Association Rules from Dynamic Class Coupling Data to Measure Class Reusability Pattern 146
Anshu Parashar and Jitender Kumar Chhabra

An Algorithm of Constraint Frequent Neighboring Class Sets Mining Based on Separating Support Items 157
Gang Fang, Jiang Xiong, Hong Ying, and Yong-jian Zhao

A Multi-period Stochastic Production Planning and Sourcing Problem with Discrete Demand Distribution 164
Weili Chen, Yankui Liu, and Xiaoli Wu

Exploration of Rough Sets Analysis in Real-World Examination Timetabling Problem Instances 173
J. Joshua Thomas, Ahamad Tajudin Khader, Bahari Belaton, and Amy Leow

Community Detection in Sample Networks Generated from Gaussian Mixture Model 183
Ling Zhao, Tingzhan Liu, and Jian Liu

Efficient Reduction of the Number of Associations Rules Using Fuzzy Clustering on the Data 191
Amel Grissa Touzi, Aicha Thabet, and Minyar Sassi

A Localization Algorithm in Wireless Sensor Networks Based on PSO 200
Hui Li, Shengwu Xiong, Yi Liu, Jialiang Kou, and Pengfei Duan

Game Theoretic Approach in Routing Protocol for Cooperative Wireless Sensor Networks 207
Qun Liu, Xingping Xian, and Tao Wu

Machine Learning Methods

A New Collaborative Filtering Recommendation Approach Based On Naive Bayesian Method 218
Kebin Wang and Ying Tan

Statistical Approach for Calculating the Energy Consumption by Cell Phones 228
Shanchen Pang and Zhonglei Yu

Comparison of Ensemble Classifiers in Extracting Synonymous Chinese Transliteration Pairs from Web 236
Chien-Hsing Chen and Chung-Chian Hsu

Combining Classifiers by Particle Swarms with Local Search 244
Liying Yang

An Expert System Based on Analytical Hierarchy Process for Diabetes Risk Assessment (DIABRA) 252
Mohammad Reza Amin-Naseri and Najmeh Neshat

Practice of Crowd Evacuating Process Model with Cellular Automata Based on Safety Training 260
Shi Xi Tang and Ke Ming Tang

Feature Selection Algorithms

Feature Selectionfor Unlabeled Data 269
Chien-Hsing Chen

Feature Selection Algorithm Based on Least Squares Support Vector Machine and Particle Swarm Optimization 275
Song Chuyi, Jiang Jingqing, Wu Chunguo, and Liang Yanchun

Unsupervised Local and Global Weighting for Feature Selection 283
Nadia Mesghouni, Khaled Ghedira, and Moncef Temani

Graph-Based Feature Recognition of Line-Like Topographic Map Symbols 291
Rudolf Szendrei, István Elek, and Mátyás Márton

Automatic Recognition of Topographic Map Symbols Based on Their Textures 299
Rudolf Szendrei, István Elek, and István Fekete

Using Population Based Algorithms for Initializing Nonnegative Matrix Factorization 307
Andreas Janecek and Ying Tan

A Kind of Object Level Measuring Method Based on Image Processing 317
Xiaoying Wang and Yingge Chen

Pattern Recognition Methods

Fast Human Detection Using a Cascade of United Hogs 327
Wenhui Li, Yifeng Lin, and Bo Fu

The Analysis of Parameters t and k of LPP on Several Famous Face Databases 333
Sujing Wang, Na Zhang, Mingfang Sun, and Chunguang Zhou

Local Block Representation for Face Recognition 340
Liyuan Jia, Li Huang, and Lei Li

Feature Level Fusion of Fingerprint and Finger Vein Biometrics 348
Kunming Lin, Fengling Han, Yongming Yang, and Zulong Zhang

A Research of Reduction Algorithm for Support Vector Machine 356
Susu Liu and Limin Sun

Fast Support Vector Regression Based on Cut 363
Wenyong Zhou, Yan Xiong, Chang-an Wu, and Hongbing Liu

Intelligent Control

Using Genetic Algorithm for Parameter Tuning on ILC Controller Design 371
Alireza rezaee and Mohammad jafarpour jalali

Controller Design for a Heat Exchanger in Waste Heat Utilizing Systems 379
Jianhua Zhang, Wenfang Zhang, Ying Li, and Guolian Hou

Test Research on Radiated Susceptibility of Automobile Electronic Control System 387
Shenghui Yang, Xiangkai Liu, Xiaoyun Yang, and Yu Xiao

Forgeability Attack of Two DLP-Base Proxy Blind Signature Schemes 395
Jianhong Zhang, Fenhong Guo, Zhibin Sun, and Jilin Wang

Other Optimization Algorithms and Applications

Key Cutting Algorithm and Its Variants for Unconstrained Optimization Problems 403
Uthen Leeton and Thanatchai Kulworawanichpong

Transmitter-Receiver Collaborative-Relay Beamforming by Simulated Annealing 411
Dong Zheng, Ju Liu, Lei Chen, Yuxi Liu, and Weidong Guo

Calculation of Quantities of Spare Parts and the Estimation of Availability in the Repaired as Old Models 419
Zhe Yin, Feng Lin, Yun-fei Guo, and Mao-sheng Lai

The Design of the Algorithm of Creating Sudoku Puzzle 427
Jixian Meng and Xinzhong Lu

Research and Validation of the Smart Power Two-Way Interactive System Based on Unified Communication Technology 434
Jianming Liu, Jiye Wang, Ning Li, and Zhenmin Chen

A Micro Wireless Video Transmission System 441
Yong-ming Yang, Xue-jun Chen, Wei He, and Yu-xing Mao

Inclusion Principle for Dynamic Graphs 449
Xin-yu Ouyang and Xue-bo Chen

Lie Triple Derivations for the Parabolic Subalgebras of gl(n, R) 457
Jing Zhao, Hailing Li, and Lijing Fang

Non-contact Icing Detection on Helicopter and Experiments Research 465
Jie Zhang, Lingyan Li, Wei Chen, and Hong Zhang

Research on Decision-Making Simulation of "Gambler's Fallacy" and "Hot Hand" 474
Jianbiao Li, Chaoyang Li, Sai Xu, and Xue Ren

An Integration Process Model of Enterprise Information System Families Based on System of Systems 479
Yingbo Wu, Xu Wang, and Yun Lin

Special Session on Data Fusion and Swarm Intelligence

A Linear Multisensor PHD Filter Using the Measurement Dimension Extension Approach 486
Weifeng Liu and Chenglin Wen

An Improved Particle Swarm Optimization for Uncertain Information Fusion 494
Peiyi Zhu, Benlian Xu, and Baoguo Xu

Three-Primary-Color Pheromone for Track Initiation 502
Benlian Xu, Qinglan Chen, and Jihong Zhu

Visual Tracking of Multiple Targets by Multi-Bernoulli Filtering of Background Subtracted Image Data 509
Reza Hoseinnezhad, Ba-Ngu Vo, and Truong Nguyen Vu

Mobile Robotics in a Random Finite Set Framework 519
John Mullane, Ba-Ngu Vo, Martin Adams, and Ba-Tuong Vo

IMM Algorithm for a 3D High Maneuvering Target Tracking 529
Dong-liang Peng and Yu Gu

A New Method Based on Ant Colony Optimization for the Probability Hypothesis Density Filter 537
Jihong Zhu, Benlian Xu, Fei Wang, and Qiquan Wang

Special Session on Fish School Search - Foundations and Application

A Hybrid Algorithm Based on Fish School Search and Particle Swarm Optimization for Dynamic Problems 543
George M. Cavalcanti-Júnior, Carmelo J.A. Bastos-Filho, Fernando B. Lima-Neto, and Rodrigo M.C.S. Castro

Feeding the Fish – Weight Update Strategies for the Fish School Search Algorithm 553
Andreas Janecek and Ying Tan

Density as the Segregation Mechanism in Fish School Search for Multimodal Optimization Problems 563
Salomão Sampaio Madeiro, Fernando Buarque de Lima-Neto, Carmelo José Albanez Bastos-Filho, and Elliackin Messias do Nascimento Figueiredo

Mining Coherent Biclusters with Fish School Search 573
Lara Menezes and André L.V. Coelho

Author Index 583

Particle Swarm Optimization: A Powerful Family of Stochastic Optimizers. Analysis, Design and Application to Inverse Modelling

Juan Luis Fernández-Martínez[1], Esperanza García-Gonzalo[1], Saras Saraswathi[2], Robert Jernigan[2], and Andrzej Kloczkowski[3]

[1] Dept. of Mathematics, Oviedo University, Spain
[2] Dept. of Biochemistry, Biophysics & Molecular Biology, Iowa State University, USA
[3] Battelle Center for Mathematical Medicine, Ohio State University, USA

Abstract. Inverse problems are ill-posed: the error function has its minimum in a flat elongated valley or surrounded by many local minima. Local optimization methods give unpredictable results if no prior information is available. Traditionally this has generated mistrust for the use of inverse methods. Stochastic approaches to inverse problems consists in shift attention to the probability of existence of certain kinds of models (called equivalent) instead of "looking for the true model". Also, inverse problems are ill-conditioned and often the observed data are noisy. Global optimization methods have become a good alternative to sample the model space efficiently. These methods are very robust since they solve the inverse problem as a sampling problem, but they are hampered by dimensionality issues and high computational costs needed to solve the forward problem (predictions). In this paper we show how our research over the last three years on particle swarm optimizers can be used to solve and evaluate inverse problems efficiently. Although PSO is a stochastic algorithm, it can be physically interpreted as a stochastic damped mass-spring system. This analogy allowed us to introduce the PSO continuous model, to deduce a whole family of PSO algorithms, and to provide some results of its convergence based on the stochastic stability of the particle trajectories. This makes PSO a particularly interesting algorithm, different from other global algorithms which are purely heuristic.

We include the results of an application of our PSO algorithm to the prediction of phosphorylation sites in proteins, an important mechanism for regulation of biological function. Our PSO optimization methods have enabled us to predict phosphorylation sites with higher accuracy and with better generalization, than other reports on similar studies in literature. Our preliminary studies on 984 protein sequences show that our algorithm can predict phosphorylation sites with a training accuracy of 92.5% and a testing accuracy 91.4%, when combined with a neural network based algorithm called Extreme Learning Machine.

Keywords: Particle Swarm Optimization, Stochastic Stability Analysis, Inverse Problems.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 1–8, 2011.
© Springer-Verlag Berlin Heidelberg 2011

1 Global Optimization Methods and Sampling

Most inverse problems can be written in discrete form as $\mathbf{d} = \mathbf{F}(\mathbf{m})$ where $\mathbf{d} \in \mathbf{R}^s$ is the observed data, $\mathbf{m} \in \mathbf{R}^n$ is the vector containing the model parameters, and $\mathbf{F} : \mathbf{R}^n \rightarrow \mathbf{R}^s$ is the physical model, that typically involves the solution of a set of partial differential equations, integral equations or algebraic systems. Given a particular observed data set $\mathbf{d}$, the inverse problem is then solved as an optimization problem, that is, finding the model that minimizes the data prediction error expressed as a certain norm $\|\mathbf{d} - \mathbf{F}(\mathbf{m})\|_p$.

The above optimization problem turns out to be ill-posed for three reasons: (a) the forward model $\mathbf{F}$ is a simplification of reality (hypothesis and numerical approximations included); (b) data are noisy and only partially sample the domain of interest; and (c) most applications of inverse modeling do not include sufficient prior knowledge to constrain the inversion. These three points cause an inverse problem to be quite different from any other kind of optimization problem since both physics and data are involved on the cost function. In addition, the prediction error landscape usually corresponds to functions having the global minimum located in a very flat and elongated valley or surrounded by many local minima, such as the Rosenbrock and Griewank functions. The type of the numerical difficulty found depends mainly on the forward functional $\mathbf{F}$, that is, the problem physics. The effect of data noise is to increase the presence of local minima and/or the size of the valley topography. Combinations of both pathologies can also occur in real problems.

Local optimization methods are not able to discriminate among the multiple choices consistent with the end criteria and may land quite unpredictably at any point in that area. These pathologies are treated through regularization techniques and the use of "good" prior information and/or initial guesses. Global optimization methods, such as genetic algorithms, simulated annealing, particle swarm, differential evolution, etc., are especially interesting because instead of solving the inverse problem as an optimization problem, they are able to sample the region of the model space containing the models that fit the observed data within a given tolerance, that is, they are able to provide information about the posterior distribution of the inverse model parameters. To perform this task they do not need in principle any prior model parameters to stabilize the inversion and are able to avoid the strong dependence of the solution upon noisy data.

Particle swarm optimization and its variants are interesting global methods since although they have not been designed to perform importance sampling, they are nonetheless able to provide a proxy for the distribution of the model parameters [5,7,8]. Also, PSO has its main advantage in being a very fast sampler and shows a good balance between exploration and convergence depending on the tuning of the PSO parameters.

2 Generalized PSO (GPSO) and PSO Family

The particle swarm algorithm [10] applied to optimization problems is very simple: individuals, or particles, are represented by vectors whose length is the

number of degrees of freedom of the optimization problem. First, a population of particles is initialized with random positions ($\mathbf{x}_i^0$) and velocities ($\mathbf{v}_i^0$). An objective function is used to compute the objective value for each particle. As time advances, the position and velocity of each particle is updated taking into account its objective function value and the objective function values of its neighbors. At time-step $k+1$, the algorithm updates positions $\left(\mathbf{x}_i^{k+1}\right)$ and velocities $\left(\mathbf{v}_i^{k+1}\right)$ of the individuals as follows:

$$\begin{aligned}\mathbf{v}_i^{k+1} &= \omega \mathbf{v}_i^k + \phi_1(\mathbf{g}^k - \mathbf{x}_i^k) + \phi_2(\mathbf{l}_i^k - \mathbf{x}_i^k),\\ \mathbf{x}_i^{k+1} &= \mathbf{x}_i^k + \mathbf{v}_i^{k+1},\end{aligned}$$

with

$$\phi_1 = r_1 a_g,\ \phi_2 = r_2 a_l,\ r_1, r_2 \in U(0,1)\ \omega, a_l, a_g \in \mathbb{R},$$

where $\mathbf{l}_i^k$ is the $i-$th particle's best position, $\mathbf{g}^k$ the global best position within the whole swarm, $\phi_1,\ \phi_2$ are the random global and local accelerations, and ω is a real constant called inertia weight. Finally, r_1 and r_2 are random numbers uniformly distributed in $(0,1)$ to weight the global and local acceleration constants, a_g and a_l.

PSO can be seen as the particular case for $\Delta t = 1$ of the generalized PSO (GPSO) algorithm [2]:

$$\begin{aligned}v(t+\Delta t) &= (1-(1-\omega)\,\Delta t)\,v\,(t) + \phi_1 \Delta t\,(g\,(t) - x\,(t)) + \phi_2 \Delta t\,(l\,(t) - x\,(t))\,,\\ x(t+\Delta t) &= x(t) + v(t+\Delta t)\Delta t.\end{aligned}$$

This algorithm can be written only in terms of position using three points ($t+\Delta t$, t and $t-\Delta t$) difference equation:

$$x(t+\Delta t) + Ax(t) + Bx(t-\Delta t) = (\phi_1 g(t) + \phi_2 l(t))\Delta t^2 \tag{1}$$

with

$$A = \Delta t(1-w)\Delta t - 2 + \Delta t^2 \phi \qquad B = 1 - (1-w)\Delta t$$

This model was derived using a mechanical analogy: a damped mass-spring system with unit mass, damping factor, $1-\omega$, and total stiffness constant, $\phi = \phi_1 + \phi_2$, the so-called PSO continuous model:

$$\begin{cases} x''(t) + (1-\omega)\,x'(t) + \phi x(t) = \phi_1 g\,(t-t_0) + \phi_2 l\,(t-t_0)\,,\ t \in R,\\ x(0) = x_0,\\ x'(0) = v_0. \end{cases}$$

In this case $x(t)$ stands for the coordinate trajectory of any particle in the swarm. The interaction with other particles in the swarm take place through the local and global attractors, $l\,(t)\,,\ g\,(t)$. In this model, mean particle trajectories oscillate around the point:

$$o\,(t) = \frac{a_g g\,(t-t_0) + a_l l\,(t-t_0)}{a_g + a_l}.$$

and the attractors might be delayed a time t_0 with respect to the particle trajectories [3].

Beginning with the mechanical analogy and the PSO continuous model, a family of PSO members having different properties with regard to their exploitation/exploration balance are derived. We can have progressive, centered and regressive discretizations for both acceleration and velocity. The result can be a three point difference equation as in PSO (1), CC-PSO, CP-PSO , RR-PSO and PP-PSO [4]; or a four point difference equation as in RC-PSO, RP-PSO, PR-PSO and PC-PSO [4].

The consistency of the different PSO family members has been related to the stability of their first and second order trajectories [3,6]. The type of mean trajectories depends on the character of the eigenvalues of the first order difference equation. Basically there are four kinds of trajectories: damped oscillatory in the complex eigenvalue region, symmetrically and asymmetrically zigzagging in the regions of negative real eigenvalues and nearlymonotonous decreasing character in the region of positive real eigenvalues. Maximum exploration in reached in the complex region. The second order trajectories show a similar kind of behavior. The second order spectral radius controls the rate of attenuation of the second order moments of the particle trajectories (variance and temporal covariance between $x(t)$ and $x(t + \Delta t)$) . These results have been confirmed by numerical experiments with different benchmark functions in several dimensions. For GPSO, CC-PSO and CP-PSO the best parameter sets (ω, a_g, a_l) are located on the first order complex region, close to the upper border of the second order stability region where the attraction from the particle oscillation center is lost, i.e. the variance becomes unbounded; and around the intersection to the median lines of the first stability regions where the temporal covariance between trajectories is close to zero. For PP-PSO, the good parameter sets are in the complex region close to the limit of second order stability and near $\overline{\phi} = 0$. The good parameters sets for RR-PSO are concentrated around the line defined by the equation $\overline{\phi} = 3\left(\omega - \frac{3}{2}\right)$, mainly for inertia values greater than two. This line is located in a zone of medium attenuation and high frequency of trajectories.

3 Selection of the PSO Version

PSO can be viewed not as a unique algorithm, but as a set of algorithms that can be used for exploitation (looking for a unique global minimum) and/or exploration (sampling the misfit region in the model space) purposes. The design of the PSO version (explorative or exploitative) depends mainly on two factors:

1. The first factor is the (ω, a_l, a_g) point that has been selected. The greatest explorative behavior is achieved when (ω, a_l, a_g) is close to or even below the second-order upper limit stability. Also, for same total mean acceleration $\overline{\phi} = (a_g + a_l)/2$, choosing $a_l > a_g$ causes the algorithm to be more explorative.
2. The second factor is the $\triangle t$ parameter, which is a numerical constriction factor to achieve stability. It is possible to show analytically that the first and second order stability regions increase their sizes and tend towards the stability region of the PSO continuous model, $\left\{\omega < 1, \overline{\phi} > 0\right\}$, when $\triangle t$ goes to zero. In this case the exploration is increased around the global best

solution. Conversely, when $\triangle t$ is greater than one, the first and second order stability regions shrink in size and the exploration is increased over the whole search space. This feature may help to avoid entrapment in local minima.

Particle swarm optimization and its variants are able to approximate the posterior distribution of the model parameters faster than Markov Chain Monte Carlo methods. To correctly perform this task a good balance between exploration and exploitation is needed, and the CP-PSO version seems to be a good option [5]. Conversely when only a good model (the candidate for a global minimum) is needed and no uncertainty analysis is performed, the RR-PSO and GPSO versions have better convergence rates to locate such a solution. Also, for example the RC-GPSO version has worse convergence for some functions but shows the best results for the Rastrigin function, thus it can be an interesting option to try instead of PSO in case of multimodal error landscapes. These facts can be taken into account to select the most appropriate PSO version when facing real problems.

4 How to Input Prior Information

It is well established in optimization theory that different norms and penalties defined in the cost function lead to different kinds of solutions with different regularity. For instance, the l_1-norm allows searching for sparse and "blocky" solutions. Conversely, soft solutions can be found using the l_2-norm, introducing as a penalty for some regularity requirements in the model.

In the PSO case, soft solutions are found in a simple way by smoothing the global best, providing to the algorithm an attractor with the prescribed smoothness [7]. Reference models can be input very easily into the PSO algorithms in at least three different ways. One way is as a particle that is introduced occasionally into the swarm. If the reference model is compatible with the data, the reference model might influence the oscillation center. A second way is through the misfit functional, including a regularization term:

$$c(\mathbf{m}) = w_1 \left\| \mathbf{d} - \mathbf{F}(\mathbf{m}) \right\| + w_2 \left\| \mathbf{m} - (\mathbf{m_{ref}}) \right\|_{p'},$$

where w_1 and w_2 are real weights. Finally, the model can be introduced as an attractor into the force team:

$$\begin{aligned} v(t+\triangle t) &= (1-(1-\omega)\,\Delta t)\, v(t) + \phi_1 \triangle t\,(g(t) - x(t)) \\ &\quad + \phi_2 \triangle t\,(l(t) - x(t)) + \phi_3 \triangle t\,(m_{ref} - x(t)), \\ x(t+\triangle t) &= x(t) + \triangle t\, v(t+\triangle t). \end{aligned}$$

$\phi_3 \triangle t\,(m_{ref} - x(t))$ might also replace the global best attractor $\phi_1 \triangle t\,(g(t) - x(t))$ and the end of the algorithm execution. In all cases, the reference model $\mathbf{m}_{ref}$ has the same dimensions as the other particles in the swarm.

5 Phosphorylation Site Prediction Using PSO

Phosphorylation is a post-translational modification added to proteins to control and regulate their activities. It is an important mechanism for regulation of biological functions. Phosphorylated sites are known to be present often in intrinsically disordered regions of proteins that lack unique tertiary structures, and thus have less information available about the structures of the phosphorylated sites. It is not always possible to detect these sites experimentally. It will be useful to have computational methods to efficiently predict these sites. It is an important challenge to predict phosphorylation sites in protein sequences obtained from high-throughput sequencing of genomes. Phosphorylation sites may aid in the determination of the functions of a protein or even differentiating mechanisms of protein functions in healthy and diseased states. Our PSO algorithm is combined with neural networks to model and predict experimentally determined phosphorylation sites in protein sequences.

5.1 Methods and Data Generation

A neural network-based Extreme Learning Machine (ELM) is used for for predicting phosphorylation sites. The parameters of the ELM are tuned by our PSO algorithm.

Extreme Learning Machine: ELM is a modified version of Single Layer Feedforward Network (SLFN) where the input weights are chosen randomly and the output weights are calculated analytically. Activation functions such as sigmoidal and Gaussian functions can be used for the hidden neuron layer, while a linear activation function is used for the output neurons. ELM is a fast and simple algorithm compared to traditional Neural Networks and is capable of finding the best results using smaller resources. If the parameters of SLFN (input weights and the bias of the hidden layer) are randomly chosen, SLFNs become a linear system in which the output weights can be determined analytically through a Moore-Penrose generalized pseudo-inverse operation of the hidden layer output matrices. This improved algorithm is called the Extreme Learning Machine. A comprehensive overview of ELM was given by Huang et al., in [9,11]. The parameters of ELM are optimized using our PSO algorithm.

Data generation for phosphorylation prediction: 13604 sequences were obtained from the Phospho. ELM database [1], where experimental phosphorylation data found in the literature has been stored for public use. In these sequences, a single residue is marked as a phosphorylated residue while all others are unphosphorylated. If there are multiple phosphorylation sites, then they are given as two separate sequences. The phosphorylated sites are usually one of three residues, namely, serine, threonine or tyrosine, but for our preliminary study we are considering only two classes where a residue is either phosphorylated or not, i.e., we do not consider the actual type of residue that is phosphorylated.

The sequences in the data are coded using a twenty digit binary code with the letter of interest being a 1 and the remaining letters denoted as a zero. A sliding

window of 9 residues are used to represent the data. This data is then used for determining whether a residue is phosphorylated or not. Since only one residue in each sequence is marked as phosphorylated, there were 13604 residues with positive class for phosphorylation but many more residues which were in the negative class for unphosphorylated residues. To maintain a balance, the same number of residues were selected from each group for the classification using ELM and PSO.

5.2 Results and Discussion

1968 residues are used for each classification of which 984 residues are phosphorylated and 984 are not phosphorylated. Similarly, the test set also has 985 residues in each class. Our PSO optimization methods have enabled us to predict phosphorylation sites with a training accuracy of 92.5%, a testing accuracy 91.4% and Mathews correlation coefficient of 0.84. In the testing results, the sensitivity for class-1 (non-phosphorylated) is 85.6% and specificity is 99.5% , with accuracy of 99.6%. For class-2 (phosphorylated) residues, sensitivity is 85.6% and specificity is 85.6% , but the accuracy was lower at 83.3%. So, the results for the negative class are higher than the results for the positive class (phosphorylated), although there were the same numbers of each class in our trials.

6 Conclusions

Particle swarm optimization (PSO) is an evolutionary computational technique used for optimization motivated by the social behavior of individuals in large groups in nature. Different approaches have been used to understand how this algorithm works and trying to improve its convergence properties for different kind of problems. These approaches go from heuristic to mathematical analysis, passing through numerical experimentation. Although the scientific community has been able to solve a wide variety of engineering problems, the tuning of the PSO parameters has still remained one of its major drawbacks. By avoiding heuristics, it can be proved that PSO can be physically interpreted as a particular discretization of a stochastic damped mass-spring system. Knowledge of this analogy has been crucial in deriving the PSO continuous model and to deduce a family of PSO members having different properties with regard to their exploitation/exploration balance. Their convergence is related to their first and second order stability regions. Numerical experiments with different benchmark functions have shown that better performing points are close to the upper limit of second order stability. In the context of inverse problems, we address the question of how to select the appropriate PSO version: the CP-PSO is intrinsically the most exploratory version and should be selected when we want to perform sampling of the posterior distribution of the inverse model parameters. Conversely, RR-PSO, CC-PSO and GPSO provide faster convergence . Four point algorithms seem also to be very promising. Finally we show the application of PSO to optimize the parameters of a neural network (ELM) to classify phosphorylated and non-phosphorylated data with quite high accuracy. We believe that our algorithm will have many applications in biological studies.

References

1. Dinkel, H., Chica, C., Via, A., Gould, C.M., Jensen, L.J., Gibson, T.J., Diella, F.: Phospho.ELM: a database of phosphorylation sites-update 2011. Nucleic Acids Res. 39, DZ61–DZ67 (2011)
2. Fernández-Martínez, J.L., García-Gonzalo, E.: The generalized PSO: a new door to PSO evolution. J. of Artif. Evol. and Appl. 2008, Article ID 861275, 15 (2008)
3. Fernández-Martínez, J.L., García-Gonzalo, E.: The PSO family: deduction, stochastic analysis and comparison. Swarm Int. 3, 245–273 (2009)
4. Fernández-Martínez, J.L., García-Gonzalo, E.: Two algorithms of the extended PSO family. In: International Conference on Evolutionary Computation, ICEC 2010, Valencia, Spain, pp. 237–242 (October 2010)
5. Fernández-Martínez, J.L., García-Gonzalo, E., Fernández-Álvarez, J.P., Kuzma, H.A., Menéndez-Pérez, C.O.: PSO: A powerful algorithm to solve geophysical inverse problems. Applications to a 1D-Dc resistivity case. J. of Appl. Geophys. 71(1), 13–25 (2010)
6. Fernández-Martínez, J.L., García-Gonzalo, E., Fernández-Álvarez, J.: Theoretical analysis of particle swarm trajectories through a mechanical analogy. Int. J. of Comp. Int. Res. 4, 93–104 (2008)
7. Fernández-Martínez, J.L., García-Gonzalo, E., Naudet, V.: Particle swarm optimization applied to the solving and appraisal of the streaming potential inverse problem. Geophys. 75(4), WA3–WA15 (2010)
8. Fernández-Martínez, J.L., Mukerji, T., García-Gonzalo, E., Suman, A.: Reservoir characterization and inversion uncertainty via a family of particle swarm optimizers. SEG Technical Program Expanded Abstracts 29(1), 2334–2339 (2010)
9. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing 70, 489–501 (2006)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings IEEE International Conference on Neural Networks, Perth, WA, Australia, vol. 4, pp. 1942–1948 (1995)
11. Saraswathi, S., Suresh, S., Sundararajan, N.: ICGA-PSO-ELM approach for accurate multiclass cancer classification resulting in reduced gene sets in which genes encoding secreted proteins are highly represented. IEEE/ACM Trans. on Comp. Biol. and Bioinforma. 8, 452–463 (2011)

Building Computational Models of Swarms from Simulated Positional Data

Graciano Dieck Kattas and Michael Small

Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
g.dieck@polyu.edu.hk, ensmall@polyu.edu.hk

Abstract. A computational method that automatically builds dynamical models of swarming systems from positional data is introduced. As an initial test for the approach, the classical Vicsek model is used to make samples for the computer algorithm and retrieve a model. Time dependent separation measures are introduced in order to characterize the dynamics of a system and then compare the behaviors of the source and retrieved model. Cases of low and high density interactions are considered to verify the generality of the models. The results show the retrieved models are capable of emulating the collective behavior well, especially when the interaction structure resembles the one of the source model.

Keywords: mathematical modeling, swarming, computer simulations, system identification.

1 Introduction

A new area of scientific research emerging is the study of the collective behavior exhibited by a group of interacting individuals, referred as swarming when applied to animal movement. The ability to obtain accurate positional data of birds from GPS and cameras has opened the door to more advanced analyses of bird flocking [1, 2]. Most approaches of mathematically modeling collective behavior using dynamical systems have involved building a model using physical laws to generate data that resembles the phenomena in question. The opposite approach is to use positional time series data from measurements to build a model that fits it as best as possible, especially using computers to automate the process. This methodology, sometimes referred as system identification, is extensively used in other areas, especially control theory and econometrics. Due to the complexity observed in collective behavior, we suggest taking this automated approach, and take advantage of computing power to automatically build nonlinear dynamical models based on the real system. In this paper, we shall consider the basic task of building models from simulated noisy data, which can be considered an initial step towards the automated modeling of real collective systems from experimental data.

Mathematical models, in particular dynamical models, have been used by biologists, physicists, and mathematicians, to illustrate animal movement or

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 9–18, 2011.
© Springer-Verlag Berlin Heidelberg 2011

interaction. Early efforts involved models with physical laws and diffusion equations [5]. Discrete time generic models of collective systems, can be used to simulate swarming with complex behavior using simple mathematical rules [3, 4].

The automatic modeling of dynamical systems has been applied in many fields. Some new techniques use data from real systems to identify nonlinear ODE models [7, 8, 9], while others focus on inferring the natural laws of physical systems [6]. Other approaches have considered generic discrete-time nonlinear models to characterize chaotic dynamics [10, 11, 12].

2 Target System: The Vicsek Model

The Vicsek model [3], is probably the simplest nonlinear model capable of exhibiting several behaviors common to swarming. The model consists of a discrete time system composed initially of several particles in a square of linear size L, each with 2D positions that are updated according to:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \tag{1}$$

The velocities consist of a constant magnitude v and its direction defined by an angle θ:

$$\mathbf{v}_i(t+1) = v \begin{pmatrix} \cos(\theta_i(t+1)) \\ \sin(\theta_i(t+1)) \end{pmatrix} \tag{2}$$

The directions of the particles are updated by averaging the components of their nearest neighbors:

$$\theta_i(t+1) = \arctan \frac{\langle \sin(\theta_i(t)) \rangle_r}{\langle \cos(\theta_i(t)) \rangle_r} + \Delta\theta \tag{3}$$

In equation 3, $\langle . \rangle_r$ denotes average of all particles within a fixed radius r of particle i (including itself) and $\Delta\theta$ is a uniformly distributed random number in the interval $[-\eta/2, \eta/2]$, symbolizing the noise of the system. The original model uses periodic boundary conditions but in order to have more realistic data, for our simulations the boundaries of the square cell were removed so that the full data can be continuous. From the different initial conditions shown in [3], we are interested in the cases of high and low density (with low noise) since they show behaviors analogous to swarming, in small and large groups respectively.

3 Radial Basis Models

It is important to clarify that any effective nonlinear modeling method such as [7, 8, 9] could be used to fit the Vicsek data to a dynamical model. We selected the discrete-time radial basis approach originally presented in [10, 11, 12] as the tool to build our models due to its proven capability of modeling a wide variety of nonlinear and chaotic behavior.

3.1 General Form

From an input scalar time series $y(t)$, the method essentially consists in building the best model of the form:

$$y(t+1) = f(\mathbf{z}(t)) + \epsilon(t) \tag{4}$$

where $\mathbf{z}(t) = [y(t), y(t-1), ..., y(t-d)]$ is the embedding of the system and $\epsilon(t)$ is the model prediction error. The former can be easily interpreted as the past values from the time series data $y(t)$ that the model will consider for calculating the prediction for $t+1$. The samples used for optimization of the model are built from time series $y(t)$ using the embedding $\mathbf{z}(t)$. The function form essentially consists of a linear term equivalent to commonly used autoregressive models (AR), and a nonlinear term consisting of a sum of nonlinear radial basis functions. Both the structure of the function and the algorithm that builds and optimizes the model, are described in detail in [10, 11]. All this optimization is done by minimizing the Minimum Description Length (MDL) [13] to prevent overfitting of the data.

3.2 Modeling and Embedding for Swarming

For the particular case of swarm modeling, we want to use the same model for each particle just like in the Vicsek model. Since we are working with positions in 2D space, the position of a particle depends on two components, which means we require two different functions for a single model. Therefore the single model that all particles follow can be defined as:

$$\mathbf{f}[\mathbf{z}(t)] = \begin{pmatrix} f_1(\mathbf{z}(t)) \\ f_2(\mathbf{z}(t)) \end{pmatrix} \tag{5}$$

To simplify the model, instead of trying to predict the absolute position $\mathbf{x}_i$ at $t+1$, we can substitute equation 4 by predicting the relative change in position $\Delta\mathbf{x}_i(t+1) = \mathbf{x}_i(t+1) - \mathbf{x}_i(t)$.

$$\Delta\mathbf{x}_i(t+1) = \mathbf{f}[\mathbf{z}_i(t)] + \mathbf{e}_i(t) \tag{6}$$

where $\mathbf{e}(t)$ is an array with the model prediction errors for each coordinate. The embedding $\mathbf{z}(t)$ of a particle i should consider enough information from the nearest neighbors, since they have a direct influence in its motion. To define our embedding $\mathbf{z}_i(t)$, we can consider the average change in position $\Delta\mathbf{x}_i(t)$ of i and its neighbors, since it gives enough information about the magnitude and direction of velocity $\mathbf{v}_i(t)$, which actually is enough to model the Vicsek data (see equations 1-3). Using an embedding which considers neighbor data instead of values of past time intervals (as in equation 4) does not affect the computer algorithm, since what it does is fit output samples to their respective instances of the embedding vector (inputs to the function). With all this in mind, the embedding is different for each particle, since each has different nearest neighbors:

$$\mathbf{z}_i(t) = \langle \Delta \mathbf{x}_i(t) \rangle_{M,i} \tag{7}$$

Of importance here, is that the neighborhood of the average in equation 8 (M) has been modified from the one in equation 3 (r). This topological average denotes averaging over particle i and a fixed number of M nearest neighbors. This significant difference in the collective interaction has been studied recently [2], and it will be considered for our models in order to provide a scenario where the structure of the source system is unknown, and it is hypothesized to use topological interaction. Practically speaking, this is not a far fetched assumption, since from several simulations of the low and high density cases (with low noise) of the Vicsek model, we have verified that the steady state number of neighbors of particles is loosely constant, especially for the low density case.

4 Measuring Swarms

When considering models for swarming, it is useful to have a measure that can characterize and illustrate the dynamical system behavior. Traditional approaches [5] have used velocity correlation functions to calculate the variance of each individual from the centroid of the population, which should be constant for a steady state swarm. In order to have a generic measure to characterize the model data, we define the average separation of individuals from the mean position of the whole population (N) at a time interval t as:

$$\delta_g(t) = \frac{1}{N} \sum_{i=1}^{N} \| \mathbf{x}_i(t) - \langle \mathbf{x}(t) \rangle_N \| \tag{8}$$

It could also important to measure the average separation in local neighborhoods instead of the whole population. This could be particularly useful in cases where swarming occurs in small groups. A slight modification to equation 9 gives us the average separation of individuals from the centroid of their local neighborhood of neighbors:

$$\delta_l(t) = \frac{1}{N} \sum_{i=1}^{N} \| \mathbf{x}_i(t) - \langle \mathbf{x}_i(t) \rangle_{M,i} \| \tag{9}$$

For some cases, a qualitative comparison between the δ's of source and model data could be more significant than a quantitative verification (e.g. shape of curves against absolute error) for emulation of collective behavior. They can also be used to observe transient and steady state properties of the system, in a way that is reminiscent of analyses commonly done in control theory.

5 Methodology

The process followed to generate a model will now be described. A single simulation of the low density with low noise case of the Vicsek model ($L = 25, \eta = 0.1$,

see [3]) is used to generate the input data $\mathbf{x}(t)$, which is a time series matrix containing the positions of both coordinates for each particle at each time interval. The method follows:

Do for each positional coordinate j (2 in total):

1. Build samples for the function f_j using as output $\Delta x_i^{(j)}(t+1)$ with embeddings (equation 8) from all particles i
2. Run the modeling algorithm
3. Set the retrieved model as f_j

Due to the randomness involved in the radial basis functions used to build the model [10, 11], it is perhaps appropriate to run the algorithm several times to retrieve several models and keep the best one. To discriminate between these retrieved models, we decided to give preference to the global behavior for this model selection, and thus we average δ_g over K simulations each with T time intervals: $\overline{\delta}_g = \langle \delta_g(t) \rangle_{K,T}$. Using this measure, the model with least absolute error $|\overline{\delta}_g^{(data)} - \overline{\delta}_g^{(model)}|$ is selected as the best model.

6 Results

After obtaining the best model following the procedure from the previous section, a set of ten simulations of both the Vicsek and the retrieved model using low density initial conditions, were used to average and calculate the $\delta_g(t)$ and $\delta_l(t)$ statistics. The number of nearest neighbors (M) considered in the retrieved model structure and for calculating $\delta_l(t)$ was selected by averaging a rough steady state number of neighbors from Vicsek simulations, and found to be $M \approx 4$. After that, the whole verification process was repeated for ten high density with low noise simulations ($L = 5, \eta = 0.1$, see [3]) and $M = 30$. This verification of using high density data for a model built from low density data was considered as extrapolation to check the generality of the model.

6.1 Case 1: Regular Vicsek Model

Figure 1(a) shows that the global separation is followed very well by the model, with a sharp increase that symbolizes movement away from the centroid. The local separation measure in figure 1(b) illustrates a convergence of individuals which later settles into a steady state. For the simulations of the source Vicsek model, a pseudo-steady state is observed with the slight increase coming at $t > 250$. In Figure 2, we can see snapshots of a simulation of both the source and the retrieved model. It shows that in the Vicsek simulation there are a few stranded particles moving away in groups of less than five, and this is why its δ_l measure in figure 1(b), which considers neighborhoods of five, has the slight increase at $t = 250$ but no such effect on the retrieved model. This can be reasoned with the fact that in the Vicsek model, the radius nearest neighbor interaction causes far away particles

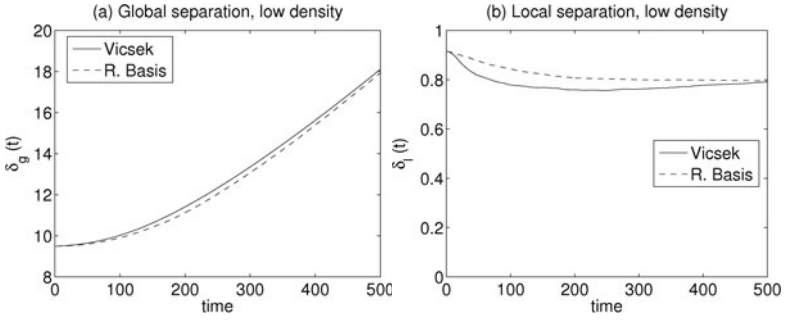


Fig. 1. Global (a) and local (b) separation measures for the low density case of the Vicsek model

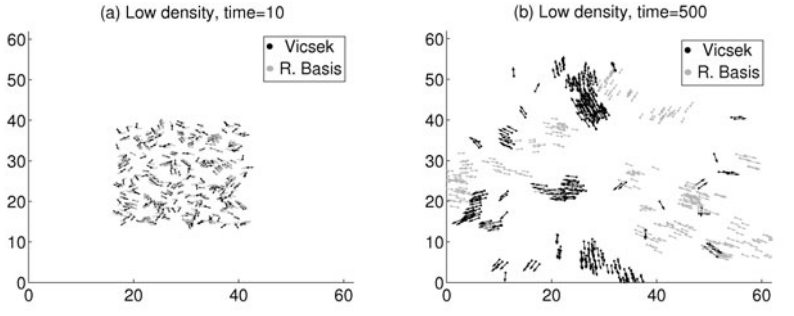


Fig. 2. Simulation snapshots of the Vicsek model and the retrieved model, for a low density case with the same initial conditions

to stay isolated, at a contrast with the structure of the retrieved model. Nevertheless, the global behavior of several groups of particles moving away in random directions, is clearly emulated by the model.

When testing the extrapolating simulations of high density initial conditions, Figure 3(a) shows how the global separation has a notable deviation. As can be observed in figure 3(b), again the local interactions settle into a steady state, with a very slight increase in the Vicsek model caused by the noise, while the retrieved model is stable. Figure 4 shows a simulation where the global deviation can be seen: the source data divides into two major swarms with a few stranded particles in the middle, while the model performs a globally stable unified movement of the whole population. This has to do with the fact that the steady state neighbors of the Vicsek model in the high density case is less constant, which can be reasoned from the fact that a higher concentration of particles within a noisy environment obviously cause more variation within a radius of interaction.

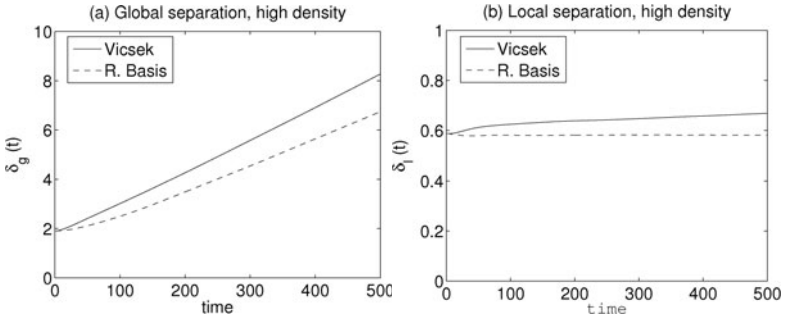


Fig. 3. Global (a) and local (b) separation measures for the high density case of the Vicsek model

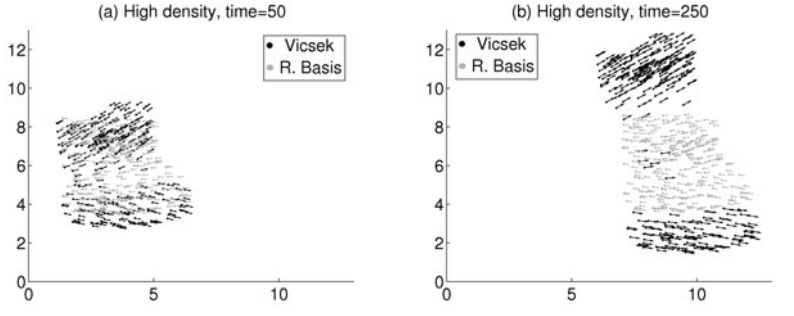


Fig. 4. Simulation snapshots of the Vicsek model and the retrieved model, for a high density case with the same initial conditions

6.2 Case 2: Modified Vicsek Model

The previous subsection showed how even with a difference in model structure, the global behavior of the low density data can be emulated adequately by the retrieved model. Nevertheless, the high density case proved to be more difficult due to the higher variation of interaction in the radius neighborhood of the Vicsek model. In this subsection, the Vicsek model will be slightly modified to consider a fixed number of neighbors in order to have equal interaction type between source and retrieved model. In formal terms, equation 3 is modified to:

$$\theta_i(t+1) = \arctan \frac{\langle \sin(\theta_i(t)) \rangle_{M,i}}{\langle \cos(\theta_i(t)) \rangle_{M,i}} + \Delta\theta \tag{10}$$

where $\langle . \rangle_M$ denotes average over the M neighbors (and itself) as explained before. The same procedure as in the previous subsection is followed to first retrieve the best model from data of the modified model and later perform simulations. In order to maintain similar global behavior to the original model, the values for

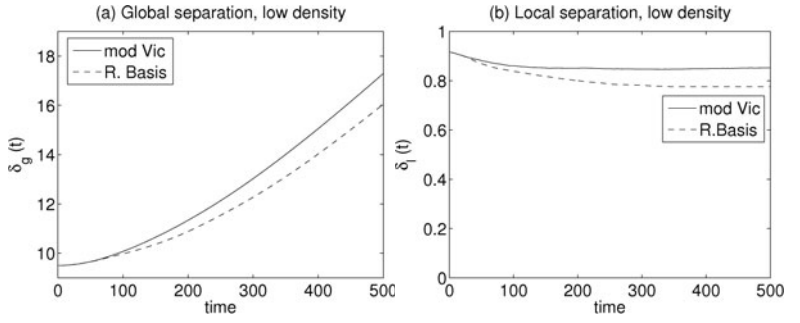


Fig. 5. Global (a) and local (b) separation measures for the low density case of the modified Vicsek model

M in the modified model were set to 4 and 30 for the low and high density cases respectively, which causes the interaction structure to be equal with the retrieved model.

For the low density simulations, Figure 5 shows how the behavior is followed qualitatively both globally and locally, just with a slight deviation. The alignment noise in the source model likely causes the separation from the data to be slightly higher than in the purely deterministic retrieved model, since this provokes small variations in the steady state orientation of the particles. Figure 6 shows how a simulation of low density initial conditions of the retrieved model follows qualitatively the behavior of the same initial conditions on the modifed Vicsek model. The behavior of the simulations are comparable with the ones obtained for the regular Vicsek model (small groups moving away), which confirms that for low densities the number of nearest neighbors within the radius of a particle is roughly constant.

The advantage of modeling this modified model can be observed with the high density data in Figure 7. The global behavior is followed much better now,

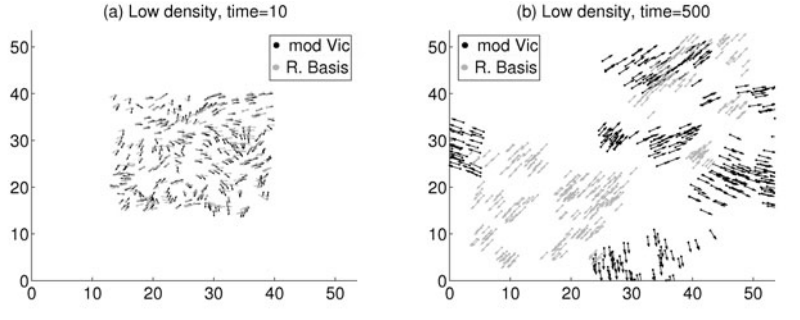


Fig. 6. Simulation snapshots of the modified Vicsek model and the retrieved model, for a low density case with the same initial conditions

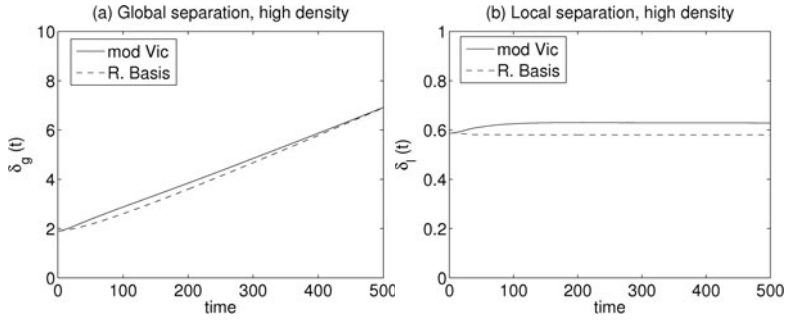


Fig. 7. Global (a) and local (b) separation measures for the high density case of the modified Vicsek model

and both local interactions qualitatively arrive at a steady state with a very low deviation. Figure 8 shows an example of how the separation of a population into two swarms is emulated very well qualitatively in a simulation.

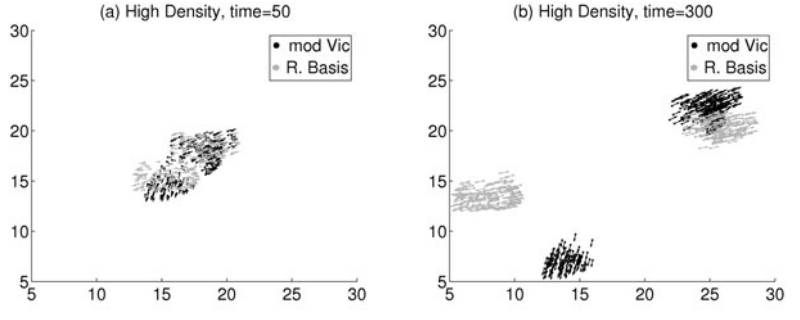


Fig. 8. Simulation snapshots of the modified Vicsek model and the retrieved model, for a high density case with the same initial conditions

7 Conclusions

The results show that automated mathematical modeling from time series data using radial basis functions can produce discrete time models capable of emulating to a certain extent the collective behavior of the classical Vicsek model. The introduced global and local separation measures over time, $\delta_g(t)$ and $\delta_l(t)$, were observed to be a very adequate representation of the behavior of the systems. From the results, we can state that for qualitative generalization of a swarming model built from data, the structure of the collective interaction in the model should resemble as much as possible the actual interaction followed in the data. This should be taken into account when taking the next step of working with real systems, in the sense that an adequate interactivity structure will be essential to retrieve good models.

Acknowledgements

GDK is currently supported by the Hong Kong PHD Fellowship Scheme (HKPFS) from the Research Grants Council (RGC) of Hong Kong.

References

[1] Nagy, M., Akos, Z., Biro, D., Vicsek, T.: Hierarchical group dynamics in pigeon flocks. Nature 464, 890–893
[2] Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Lecomte, V., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., Zdravkovic, V.: Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. Proc. Natl. Acad. Sci. USA 105(4), 1232–1237 (2008)
[3] Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., Shochet, O.: Novel type of phase transition in a system of self-driven particles. Phys. Rev. Lett. 75, 1226 (1995)
[4] Reynolds, C.: Flocks, Herds, and Schools: A Distributed Behavioral Model. In: Stone, M.C. (ed.) Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1987), pp. 25–34. ACM, New York (1987)
[5] Okubo, A., Levin, S.: Diffusion and Ecological Problems, 2nd edn., ch. 7. Springer, Heidelberg (2000)
[6] Schmidt, M., Lipson, H.: Distilling Free-Form Natural Laws from Experimental Data. Science 324(5923), 81 (2009)
[7] Bongard, J., Lipson, H.: Automated reverse engineering of nonlinear dynamical systems. Proc. Natl. Acad. Sci. USA 104(24), 9943–9948 (2007)
[8] Gennemark, P., Wedelin, D.: Efficient algorithms for ordinary differential equation model identification of biological systems. IET Syst. Biol. 1(2), 120–129 (2007)
[9] Dieck Kattas, G., Gennemark, P., Wedelin, D.: Structural identification of GMA models: algorithm and model comparison. In: Proceedings of the 8th International Conference on Computational Methods in Systems Biology, pp. 107–113. ACM, New York (2010)
[10] Judd, K., Mees, A.: On selecting models for nonlinear time series. Physica D 82, 426–444 (1995)
[11] Small, M., Judd, K.: Comparisons of new nonlinear modeling techniques with applications to infant respiration. Physica D 117, 283–298 (1998)
[12] Small, M., Tse, C.K.: Minimum description length neural networks for time series prediction. Physical Review E 66, 066701 (2002)
[13] Rissanen, J.: Stochastic Complexity in Statistical Inquiry. World Scientific, Singapore (1989)

Robustness and Stagnation of a Swarm in a Cooperative Object Recognition Task

David King and Philip Breedon

The College of Architecture Design and the Built Environment,
Nottingham Trent University, NG1 4BU
{David.King,Philip.Breedon}@ntu.ac.uk

Abstract. Swarm intelligent, cooperative object recognition forms part of cooperative construction research. A simulation model was designed and utilised to assess the suitability of a swarm of agents to identify and collect different objects, termed the Simplified Hexagonal Model. An agent in this system cannot assess different object types alone. Key to the efficiency of the system is avoiding stagnation whilst maintaining robustness. This paper examines the energy efficiency of the system when the probability of an agent moving away from an object it is trying to identify is varied. The probability of an agent moving away from an unidentified object per time-step was varied from 1:12 to 1:400. Both low and high probabilities increased the energy required to complete the task. This was more pronounced when using fewer agents. The reduced chance that the required number of agents were surrounding the same objects at the same time caused the increase.

Keywords: Swarm intelligence, cooperative object recognition, stagnation.

1 Introduction

Due to their robustness and scalability using multiple simple robotic agents for construction is a widely accepted concept [1,2,3,4,5]. The behaviours used are often inspired by those in social insect nest building [6,7,8]. Research in autonomous cooperative construction frequently involves the use of homogeneous blocks. These blocks can be attached to each other in any order to complete the construction. It is suggest in [3] that these blocks could represent prefabricated pods, containing living quarters, research areas or kitchens. More often the blocks are considered the most basic construction component. Blocks that alternate in colour but are otherwise identical are used to form walls in [2]. There are also smart blocks which act as other agents and hold information about the overall structure [5]. Perhaps another way to construct would be to use component pieces. These would take less space to transport than prefabricated pods but are more task specific than simple blocks. In this situation the component pieces would need identifying, matching and moving together. These tasks could all be completed through swarm intelligent control. This paper forms part of the research that focuses on the identification of different object shapes.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 19–27, 2011.
© Springer-Verlag Berlin Heidelberg 2011

In [9][10] and [11] each robot or agent in the swarm could individually assess and identify any object that required moving cooperatively. What if an individual agent cannot identify the object alone, due to similarities in different components? There would then be a need for cooperative sensing, where agents share partial knowledge of the components. A cooperative sensing task is discussed in [12]. This scenario used agents with different sensing capabilities deviating from the homogeneous swarm robot model, something that is important in keeping the system robust where failures may occur. In [13] given the position of the homogeneous agents and their mounted cameras they are able to construct a spatial section and assess the object type.

The authors' research aims to understand how agents, with limited sensing capabilities and no long term memory are capable of identifying different types of objects that an individual cannot assess alone. To do this the agents react to their local surroundings by changing their state behaviour and display colours. This paper focuses specifically on the issue of avoiding stagnation when a swarm is identifying different objects where only a certain type is required to be collected to complete the task. Although this is an over simplification of real world cooperative construction tasks the model will provide useful data and control strategies that will aid the development of cooperative object recognition for cooperative construction in the future.

2 Simplified Hexagonal Model

The simplified hexagonal model (SHM) was developed in Processing [14] an open source graphical programming platform. It was constructed from a two-dimensional hexagonal grid. Each active hexagonal cell can contain, part of a larger object, an hBot agent, a boundary region or nothing. The SHM provides a quick and simple method of testing early ideas in swarm intelligence control. Information, data and control techniques gained from the model can then be transferred to more complex simulations and hardware platforms.

Agents used in the SHM are hBots, hexagonal robots. An hBot occupies one cell on the hexagonal grid and when not in contact with an object performs a random walk. Each hBot has a limited sensor range. It is aware of the condition of each of its six surrounding cells and the twelve that surround those. The hBots behave based on their state which is determined by their surrounding conditions. In the simulation they communicate their current state by changing display colour, as it is clearer to observe.

Objects are approximations of triangles or hexagons built from bound neighbouring cells. Objects can be moved if enough hBots are pushing or pulling it. The orientation and the position of the hBots around the objects do not affect their ability to move it. All the hBots that are trying to move the object move with it as a group. However, any hBots that are in the way and have not identified the object stop its and the other hBots' movement. An hBot in contact with an object will have a number of edges touching it. In figure 1 the numbers of edge contacts for each surrounding cell are shown for both triangular objects and hexagonal objects. Table 1 shows the different patterns of edge contact that three neighbouring cells make for both objects.

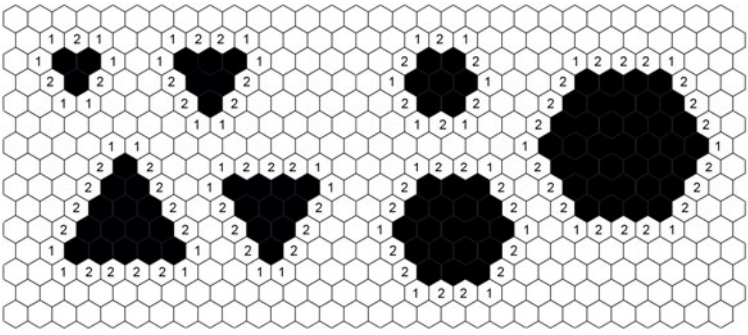


Fig. 1. Objects with surrounding cells showing number of sides in contact with object

Table 1. Patterns of sides in contact with objects with three hBots in contact with each other

Triangle: 3 hBot combinations			Hexagon: 3 hBot combinations		
Left	Centre	Right	Left	Centre	Right
1	(1)	2	-	-	-
1	(2)	1	1	(2)	1
1	(2)	2	1	(2)	2
2	(1)	1	-	-	-
-	-	-	2	(1)	2
2	(2)	2	2	(2)	2

The patterns that only occur when three hBot agents surround a triangle or hexagon but not the other in table 1 are as follows:

- Patterns 1(1)2 and 2(1)1 only occur with a triangular object.
- Pattern 2(1)2 only occurs with a hexagonal object.

With this information it was possible to produce a system where groups of hBots can identify an object as hexagonal or triangular, when there are three adjacent hBots at a corner of a shape. Table 2 explains how the hBots current state, behaviours and display colours change based on their current state and surrounding parameters.

Table 2. The state, sensed condition and behaviours that control the hBot

State	Sensed Condition	Behaviour (per time-step)	Colour
0	Searching for object or in contact with the edge of arena or collection zone.	Perform random walk avoiding collisions	Grey
1	In contact with object on one side only.	A probability to perform random walk avoiding collisions.	Green
2	In contact with object on two sides.		Blue
3	(a) In state 1 with two neighbouring hBots, one in state 1 the other state 2. (b) In (state 1 or state 2) and in contact with a state 3 hBot.	Attempt to push or pull object one cell towards goal. Goal for valid objects is the collection zone. Goal for invalid objects is outside the search area.	Red
4	(a) In state 1 with two neighbouring hBots, both in state 2. (b) In (state 1 or state 2) and in contact with a state 4 hBot.		Purple

An example of a group assessing a triangular object, the assessment of a hexagonal object is very similar, can be seen in figure 2 which shows:

a) Three hBots in contact with object in states 1 for single side contact and state 2 for dual object side contact, three hBots in state 0 are approaching object.
b) A group of three hBots form at a corner of the triangle with states 1(1)2.
c) The centre hBot of the first group of three changes to state 3 (state 3 condition a), the second group of three hBots is in contact with the object.
d) The two outer hBots change to state 3 (state 3 condition b), the second group remain in states 1(2)2 as this is common to both objects types.

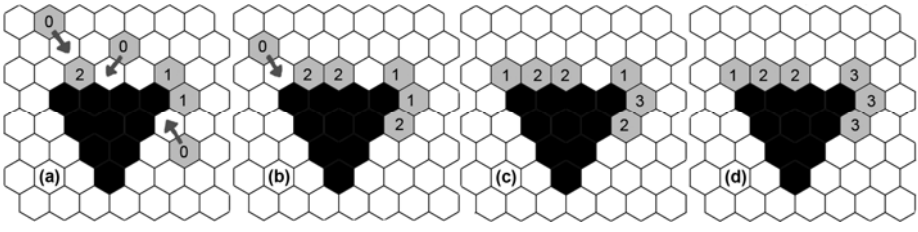


Fig. 2. Example of hBots using state behaviour to identify a triangular object. hBots in state 3 are aware they are in contact with a triangular object.

The overall program, figure 3, has three essential parts, the hBots, the objects and the arena. The arena maintains a record of what is happening in each of its cells, a hBot in a certain state, part of an object or nothing at all, then displays the information on the screen. The hBots each sense their surroundings based on the data retained in the cells around them, then use this data to select their state behaviour and display

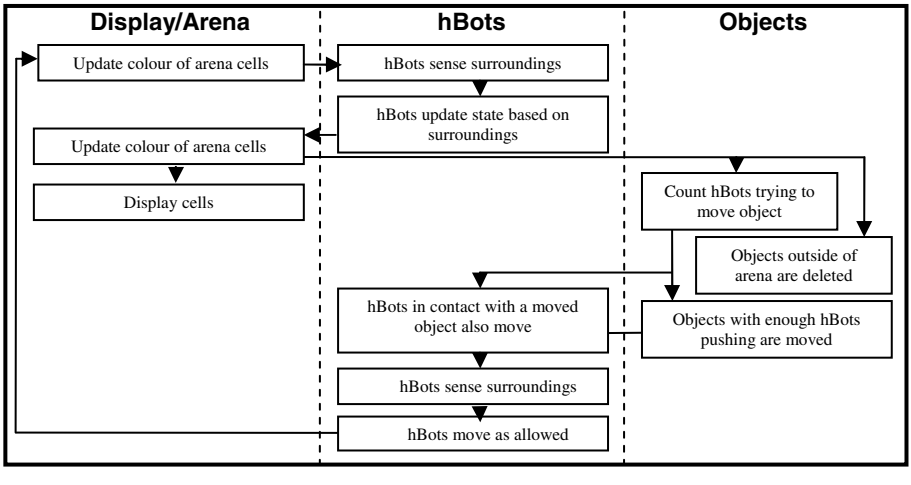


Fig. 3. The flow of the main program, broken down into the three main component parts, the updating and displaying the cells of the arena, the hBots sensing state and movement control and the objects movement and deletion

colour, which affects how they can move. Finally the objects react when enough hBots of the correct state are trying to move them and nothing is in their path and move accordingly. Any objects pushed out of the arena or into the collection zone are deleted. The decision tree, figure 4, expands on the section regarding how the hBots select their current state based on their sensed surroundings.

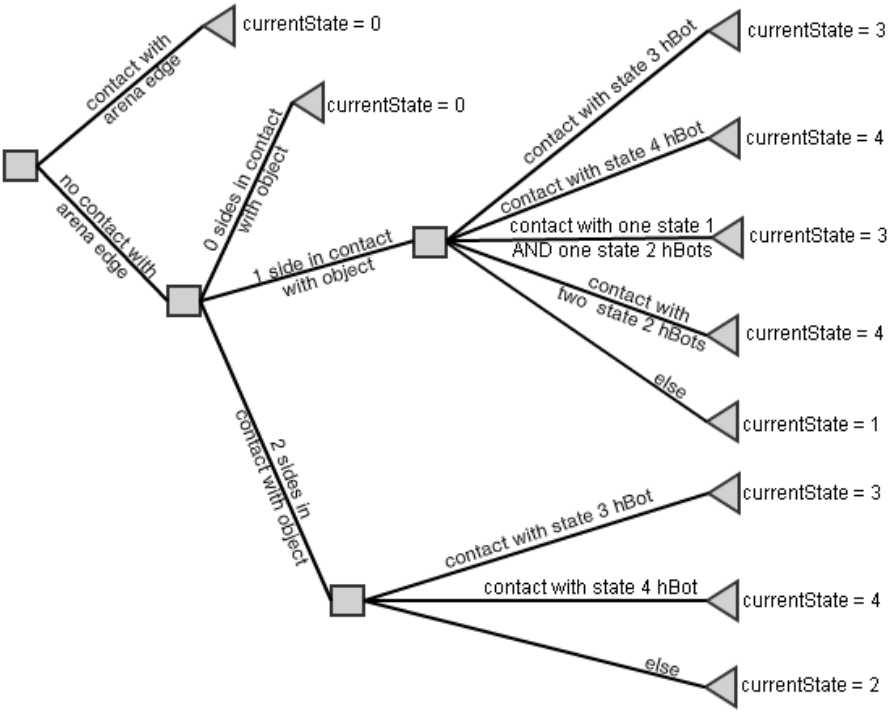


Fig. 4. A decision tree explaining how the current states of the hBots are determined based on their sensed surroundings

3 Results

An arena was constructed in a hexagonal arrangement measuring 41 cells diagonally from corner to corner. The hexagonal collection zone was placed in the exact centre measuring 11 cells diagonally from corner to corner. In figure 5 three valid 7 cell hexagon objects and three invalid 6 cell triangles are shown spaced evenly at equal distances away from the collection zone. The hBots need to collect the three valid objects from the arena to complete the task. Any invalid objects are possibly removed from the arena by the agents if they are identified.

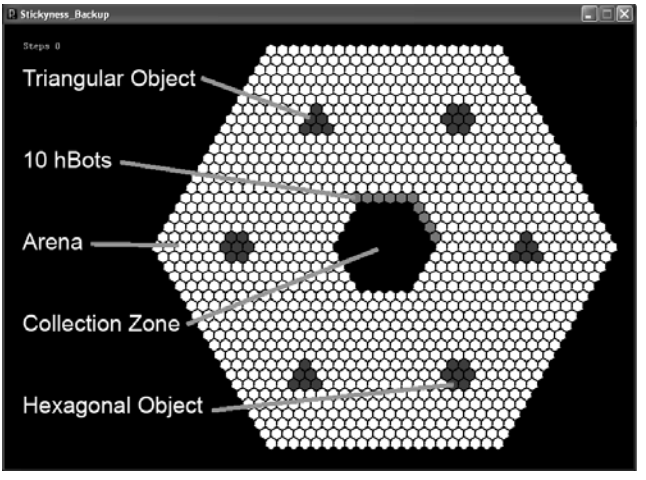


Fig. 5. Screen grab of SHM Arena at time-step 0

In order to reduce the chance of stagnation in the system any hBots that are in contact with an object but have not identified it (states 1 or 2) have a probabilistic chance each time-step to move away from the object. If this were not the case there could be situations where the hBots are spread amongst the objects without the required three in a row to identify the object. This variable is considered to be the 'stickiness' of the agent. For each time-step every hBot in states 1 or 2 has a 1 in 'stickiness' chance of moving. The probability of the hBot moving away decreases with larger values of the variable. This paper measured the affect on the energy cost of different populations of hBots when their 'stickiness' was varied from 12 to 400. The total energy cost is found by multiplying the number of hBots by the number of time-steps as in equation 1. This provides a more accurate evaluation of the systems efficiency than measuring time-steps alone. In the SHM the energy supply is currently modelled as infinite as it provides valuable information on the overall efficiency of the system as variables are adjusted rather than a fully realistic simulation of robotic agents in the field.

$$\text{time-steps} \times \text{number of hBots} = \text{energy cost} . \quad (1)$$

The simulations for each value of 'stickiness' were run with a varying hBot populations. For each number of hBots, from 10-30 at increments of 5, the experiment was run 50 times. To complete the task, three of the hexagonal objects needed to be moved by the hBots to the collection zone. The unwanted triangular objects, once identified, could be moved out of the search arena to avoid re-assessment. It required four hBots working together to move an object. This emphasised the difference in the two tasks, identifying and movement. The amount of time-steps it took to complete the tasks was recorded and the energy cost calculated.

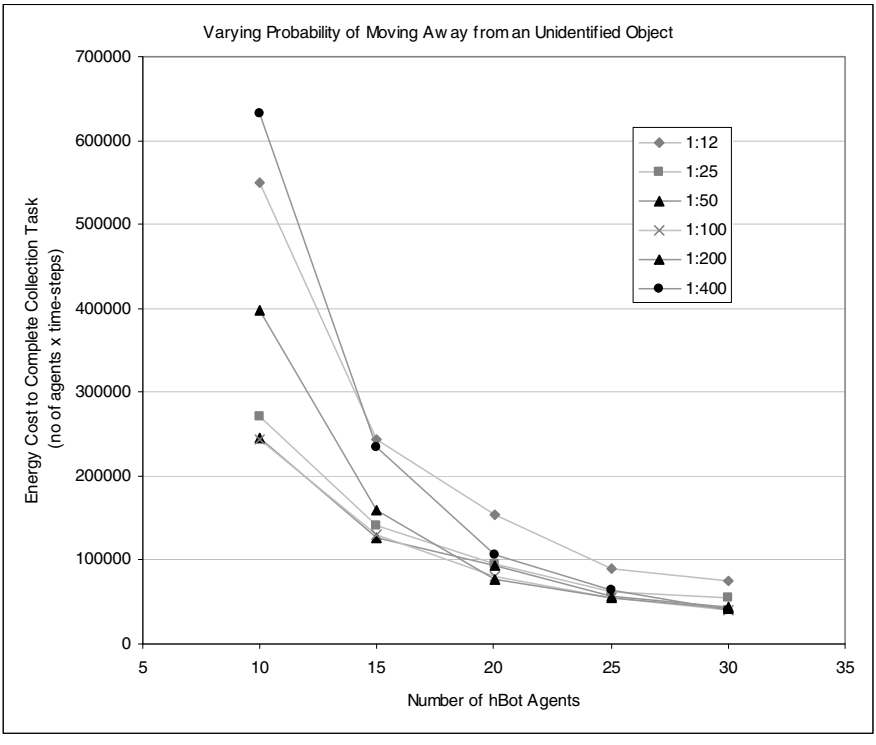


Fig. 6. The total energy cost to complete the task decreases as the hBot population increases

As in previous experiments [15] increasing the agent population in an arena of this size decreases the amount of energy used by the agents. The results presented in figure 6 show for all values of 'stickiness' there is an initial rapid decrease in the amount of energy consumed which starts to level off as the population is increased. There is a more pronounced affect when the values are at the extremities. The low probability of movement 1:12 and the high probabilities 1:200 and 1:400 have the highest energy costs. The energy cost for 1:50 and 1:100 are almost exactly the same. Overall as the number of hBot agents increases the energy cost for completing the task tend towards similar values across the board.

The variation in energy cost against the increasing stickiness is more clearly visible in figure 7. For every number of hBots used in the simulation the same pattern is seen. Similar to a 'tick' shape, the energy cost starts high when there is a 1:12 probability of moving, drops rapidly. This drop ranges from approximately 30-50% depending on the number of hBots. Between probabilities 1:50 and 1:100 the energy cost remains approximately level, variations between 1-15%. There is then a steady increase in energy consumed to complete the task from the 1:50 probability to 1:400. The variation in the energy consumed decreases as the number of hBots increases. This is because with a limited number of objects to find and identify the hBots are more likely to cluster together in the required groups of three to identify an object.

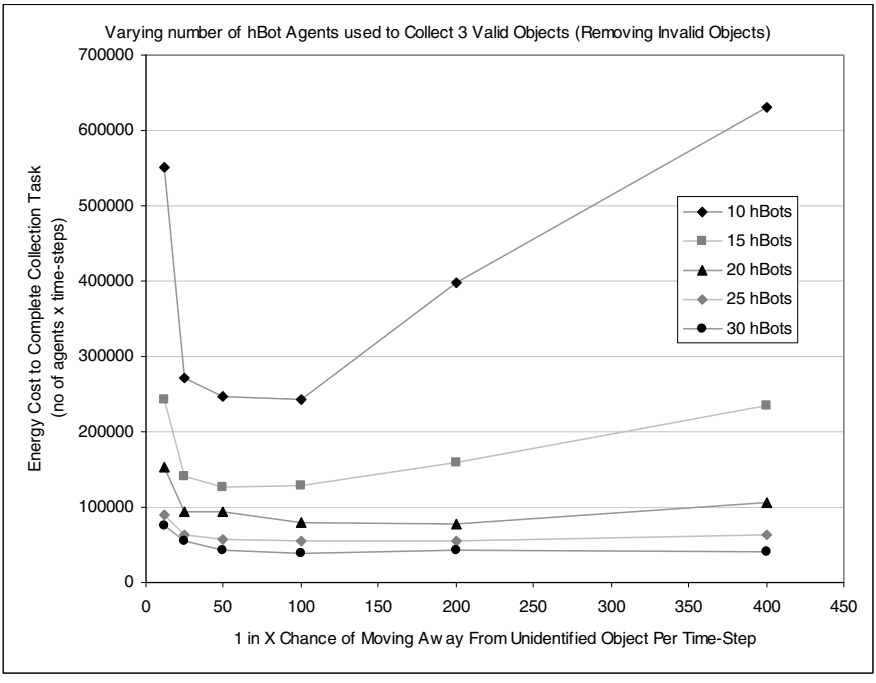


Fig. 7. The energy cost drops suddenly and rises gradually as the probability of moving away from an unidentified object per time-step decreases from 1:12 to 1:400

In one simulation (15 hBots, stickiness of 200) the task was not completed due to complete stagnation. One of the hexagonal items was collected on the 5767th time-step leaving two hexagonal and three triangular objects. The 15 hBots spread evenly and identified the five remaining objects. As the experiment requires four hBots pushing/pulling an object to move it, the task stagnated.

4 Conclusion

The ability for multiple agents to identify different objects cooperatively is an important aspect of autonomous cooperative construction. The aim of this paper was to gain an understanding of this issue, specifically where a single agent could not identify different object shapes alone. The SHM has proven to be a valuable tool for gathering a large amount of data quickly to aid in the planning for future research. It is clear that this model could not only be used for other cooperative identification research, but for other types of early swarm intelligence and swarm robotic research.

The focus of this paper was the balance between robustness and stagnation. A variable was set in the system which gave each agent a given probability to remain next to an object that it was attempting to identify. Increasing the ‘stickiness’ value reduces the probability of it moving away. For low values of this variable there were high energy costs. This represents a less robust system with a low probability of total or

temporary stagnation. However, increasing the variable too high also required a high energy cost as temporary stagnation was far more common. The situation where true stagnation occurred would not have necessarily been avoided by a different 'stickiness' value. In this specific case there were five objects in the system and three hBots on each object which had, as a group, identified the objects. In these experiments it required four hBots who are all aware of the object type in order to move it. This could be avoided by adding a second variable for the situations where agents have identified an object. Giving them a chance to move away and perhaps help another group move an object as required.

Further research using the SHM is needed to understand the limitations and capabilities of simplistic agents to cooperate to identify objects. The research will provide insight of the relevant strategies required for implementation with physical robots.

References

1. Brooks, R.A., Maes, P., Mataric, M.J., et al.: Lunar Base Construction Robots. In: IEEE International Workshop on Intelligent Robots and Systems, vol. 1, pp. 389–392 (1990)
2. Wawerla, J., Sukhatme, G.S., Mataric, M.J.: Collective Construction with Multiple Robots. In: Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2696–2701 (2002)
3. Werfel, J., Bar-Yam, Y., Nagpal, R.: Building Patterned Structures with Robot Swarms. In: Su, D., Zhang, Q., Zhu, S. (eds.) 19th International Joint Conference on Artificial Intelligence, IJCAI 2005, pp. 1495–1502 (2005)
4. Werfel, J.: Robot Search in 3D Swarm Construction. In: 1st. International Conference on Self-Adaptive and Self-Organizing Systems, pp. 363–366 (2007)
5. Werfel, J., Nagpal, R.: Three-Dimensional Construction with Mobile Robots and Modular Blocks. Int. J. Robot. Res. 27(3-4), 463–479 (2008)
6. Franks, N.R., Deneeubourg, J.L.: Self-Organizing Nest Construction in Ants: Individual Worker Behaviour and the Nest's Dynamics. Anim. Behav. 54, 779–796 (1997)
7. Theraulaz, G., Bonabeau, E.: Modelling the Collective Building of Complex Architectures in Social Insects with Lattice Swarms. J. Theor. Biol. 177, 381–400 (1995)
8. Mason, Z.: Programming with Stigmergy: Using Swarms for Construction. In: Standish, R.K., Bedau, M.A., Abbas, H.A. (eds.) Artifical Life VIII, pp. 371–374. MIT Press, Cambridge (2002)
9. Kube, C.R., Zhang, H.: Collective Robotics: From Social Insects to Robots. Adapt. Behav. 2(2), 189–219 (1993)
10. Kube, C.R., Bonabeau, E.: Cooperative Transport by Ants and Robots. Robot. Auton. Sys. 30, 85–101 (2000)
11. Zhang, D., Xie, G., Yu, J., et al.: Adaptive Task Assignment for Multiple Mobile Robots via Swarm Intelligence Approach. Robot. and Auton. Sys. 55, 572–588 (2007)
12. Payton, D., Estkowski, R., Howard, M.: Compound Behaviours in Pheromone Robotics. Robot. Auton. Sys. 44, 229–240 (2003)
13. Oswald, N., Levi, P.: Cooperative Vision in Multi-Agent Architecture. In: Del Bimbo, A. (ed.) ICIAP 1997. LNCS, vol. 1310, pp. 709–716. Springer, Heidelberg (1997)
14. Processing, `http://processing.org/`
15. King, D., Breedon, P.: Towards Cooperative Robotic Swarm Recognition: Object Classification and Validation. In: 3rd International Conference on Advanced Design Manufacture. Key Eng. Mat, vol. 450, pp. 320–324. Trans Tech Publications (2010)

Enforced Mutation to Enhancing the Capability of Particle Swarm Optimization Algorithms

PenChen Chou and JenLian Chen

Electrical Engineering Department., DaYeh University, 168, College Road,
41000 ChungHwa County, Taiwan
choup@tcts.seed.net.tw, R9903027@mail.dyu.edu.tw

Abstract. Particle Swarm Optimization (PSO), proposed by Professor Kennedy and Eberhart in 1995, attracts many attentions to solve for a lot of optimization problems nowadays. Due to its simplicity of setting-parameters and computational efficiency, it becomes one of the most popular algorithms in optimizations. However, the discrepancy of PSO is the low dimensionality of the problem can be solved. Once the optimized function becomes complicated, the efficiency gained in PSO degradates rapidly. More complex algorithms on PSO required. Therefore, different algorithms will be applied to different problems with difficulties. Three different algorithms are suggested to solve different problems accordinately. In summary, proposed PSO algorithms apply well to problems with different difficulties in the final simulations.

Keywords: Optimization, Optimization Benchmark, Particle Swarm Optimization, Genetic Algorithm, Mutation.

1 Introduction

Since 1955, Professor Kennedy and Eberhart introduced the bright new optimization algorithm called Particle Swarm Optimization (PSO); optimization algorithms are more advanced and efficient than that before [1-7, 9]. Take an example, Genetic Algorithm (GA) is another optimization one based on evolution principle with high computational burden [8].

Adjustments on parameters of PSO are under studying by many researchers all around the world. As mentioned in [9], many improved PSOs have been promoted to enhance the capability of original PSO, but fail to find the optimal solution when the dimensionality of the problem is high. In [9], arithmetic mutation borrowed from GA with low mutation rate can improve the problems with many variables. The algorithm is therefore called Improved PSO (IPSO) later. Furthermore, with enforced one-variable mutation besides GA mutation, the dimensionality can increase from 200 to 2000 for easy problems. As for the tough problem, the dimension of variables can modify from 16 to 28 with good success rate [9, 10]. This is the contribution of this paper.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 28–37, 2011.
© Springer-Verlag Berlin Heidelberg 2011

2 PSO Introduction

The formulas recommended by [1] are listed in equation (1) and (2).

$$V_i(k+1) = \omega * V_i(k) + c_1 * \varphi_1 * \left(P_i(pbest) - P_i(k)\right) + c_2 * \varphi_2 * (P(gbest) - P_i(k)) \quad . \tag{1}$$

$$P_i(k+1) = randxx * P_i(k) + V_i(k+1) \quad . \tag{2}$$

Where V stands for velocity, P stands for particle/position, pbest is the best parameter in the i-th generation, and gbest is the best solution variable array from beginning to the present generation so far. φ_1 andφ_2 are two uniform random numbers. i is the generation number. In [9, 10], the best settings of parameters in PSO with GA mutation rate of 0.0035 are

In the original PSO, c_1=c_2=2, ω=1, randxx=1, but in the modified PSO, parameters are changed. Simulation results for benchmark problems with modified PSO can be found in [3, 4, 9, 10].

3 Enforced Mutation in PSO (EMPSO) Algorithm

From the original PSO viewpoint, the adjusting number of parameters is quite few (e.g. c_1, c_2, ω, randxx). It is okay to most problems with low dimensions. In [9], GA mutation of rate 0.0035 is appended behind the original PSO algorithm. This is IPSO in [9, 10]. It is shown that the dimension for those problems can be increased further. When the dimension is going higher and higher, IPSO also showed the incapability to find the true optimum in the search. A momentum is still needed to let PSO solutions jumping out to the final global optimum from local optimum. This momentum goes to the idea that after GA mutation process, an enforced mutation mechanism is included in certain specified time interval in the whole generations. Without this enforced mutation, at the end of generation run, the best solution maybe is the global one as desired or trapped in the local optimum. The idea for creating EMPSO is that part of particles (solutions) are replaced by the global optimum sets and one variable in every set are forced to modify to a new allowable value by single mutation. Therefore, in the algorithm executing process, randomly select one variable to do the other single arithmetic mutation where other variables are adhered to the global optimal record so far (gbest). The modified generalized algorithm (EMPSO) has steps in sequences and explained as

(1) Perform the regular PSO update process as usual. Proper parameter settings in PSO for each sample problem are shown later. Equations are the same as Equation (1) and (2).
(2) GA mutation with rate of 0.0035 for sample problems with index of difficulty of 1 or 2 (defined later), but rate is increasing with running generation from 0.0035 to 0.01 for problems with index of difficulty of 3. As the rate of 0.0035, if there are 200 particles (population) in each update step, 3.5% of 200 or 7 particles (in average) are selected for mutation. 7 particles are

randomly picked up in the process. If the dimension of each particle is 200, then 7 variables out of 200 ones are under mutation process. Mutation is a kind of arithmetic mutation, that is, randomly selected a legal value between upper bound and lower bound of each selected variable to replace the original value. This is a true GA mutation usually used in GA process.

(3) After completing the step (2), the gbest may locate at the true optimal values, while other time the gbest solution is trapped in the local optimal locus. There are no any progresses in the solution search to the final. In order to overcome the trapping problem, enforced mutation is executed 3 times in the proper located intervals. In the whole generations, 3 short intervals are arranged to do the enforced mutation as shown in Fig. 5. In Fig. 5, only two enforced mutation processes are executed and shown accordingly because after completing the second enforced mutation, the true optimal solution is already found shortly later and the next generation process is stopped. Population of one-variable enforced mutation is one tenth of total population (called EM_population). gbest is copied to each randomly selected particle in EM_population. One variable is randomly selected again for each randomly selected particle. This variable is forced to obtain a new legal value in the selected bounds (All other variable values are not changed as gbest assigned). The main function of one variable mutation is to gain the ability of particles trapped in the local optimal region to jump out of the trapped region. Without this mechanism, one can not use high dimension of variables for each sample problems.

In summary, in EMPSO, the first step is to apply eq.(1) and (2) to update particle and velocity. Parameters are set on the condition of index of difficulty. GA mutation is executed with low mutation rate such as 0.0035 or from 0.0035 to 0.01. This is the second step. In the last step, enforced one-variable mutation is inserted in the generation interval (3 times in the generation run). It is shown that in the following tables that with this enforced mutation used, the dimension of variables of each sample problems can be increased mildly.

4 Sample Examples

In this section four benchmark optimization problems are tested under proposed EMPSO algorithm with enforced mutation mechanism included. Three difficulty indices (1, 2, and 3) are used to classify these four problems with small number (such as 1) to represent a relatively easy problem, while larger number (such as 3) to represent tougher one.

4.1 Example 1: Sphere Function with Difficulty Index of 1

$$f_1 = \sum_{i=1}^{n} x_i^2 \quad . \tag{3}$$

where $-100 < x_i < 100$, $i = 1,2,....,n$, n is the dimensionality or number of variables to be optimized. The objective is to find the minimum of f_1 function and the

related variable locations. Fig.1 shows the surface plot of the sphere function in 2 variables. There is a unique minimum point in the figure with location [0, 0]. Population size is 600. Parameters for PSO and enhanced mechanism (EMPSO) for this simple problem are (Algorithm #1)

(1). $c_1=c_2=1.9$, $\omega=0.763$, randxx=-10^{-4}.
(2). GA mutation rate is set to 0.0035.
(3). One-variable enforced mutation is used.

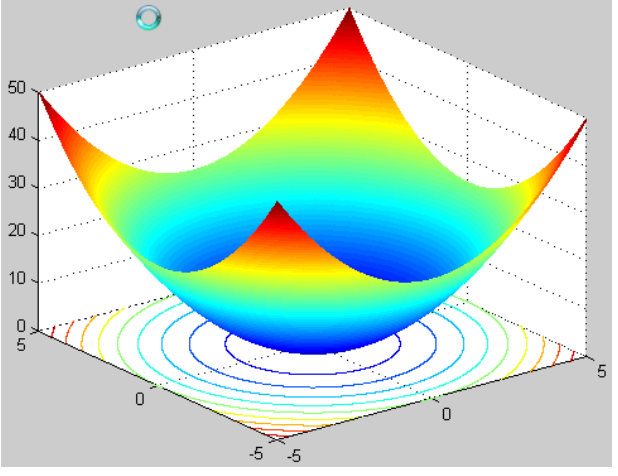


Fig. 1. Surface plot of the sphere function in n=2 variables

Table 1. Optimization results of sphere function

Number of variables (dimension)	Success rate	Generations to complete (average)	Mutate rate
2	50/50	13	0.0035
10	50/50	28	0.0035
25	50/50	36	0.0035
50	50/50	41	0.0035
200	50/50	46	0.0035
500	50/50	48	0.0035
1000	50/50	49	0.0035
2000	50/50	51	0.0035

From Table 1, the EMPSO shows its great capability to find the optimal solution with so few generations for the high dimension sphere functions with full success rates. In [9], the largest dimension is restricted to 200.

4.2 Example 2: Rastrigin Function with n Variables and Difficulty Index of 1

$$f_2 = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \quad . \tag{4}$$

Where $-100 < x_i < 100$, $i = 1,2,....., n$, n is the dimensionality or number of variables to be optimized. The objective is to find the minimum of f_2 function and the related variable locations. Fig. 2 shows the surface plot of the function in 2 variables. There is a unique minimum point with value of zero in the figure with location (0, 0). Population size is 600. Parameters for PSO and enhanced mechanism (EMPSO) are

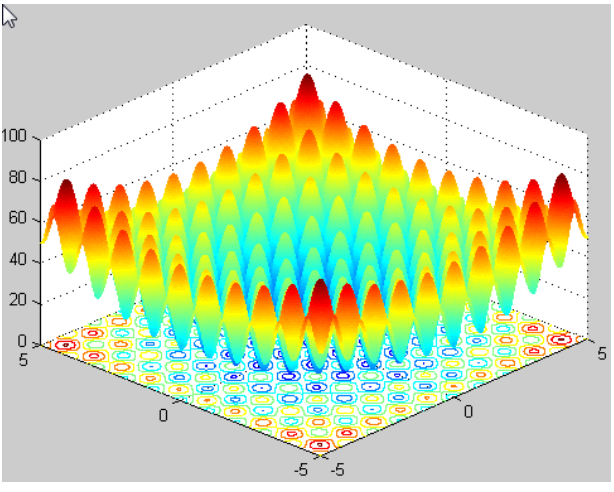


Fig. 2. Surface plot of the Rastrigin function in n=2 variables

Table 2. Optimization results of Rastrigin function

Number of variables (dimension)	Success rate	Generations to complete (average)	Mutate rate
2	50/ 50	18	0.0035
5	50/ 50	60	0.0035
10	49/ 50	238	0.0035
15	47/ 50	41	0.0035
20	45/ 50	228	0.0035
25	45/ 50	50	0.0035
30	46/ 50	130	0.0035
35	47/ 50	78	0.0035
40	48/ 50	50	0.0035
50	47/ 50	51	0.0035
200	50/ 50	56	0.0035
500	50/ 50	58	0.0035
1000	48/ 50	59	0.0035
2000	50/ 50	60	0.0035

(Algorithm #1)

(1). $c_1=c_2=1.9$, $\omega=0.763$, randxx=-10^{-4}.
(2). GA mutation rate is 0.0035.
(3). One-variable enforced mutation is used.

From Table 2, the EMPSO shows its good capability to find the optimal solution with so less generations for the high dimension Rastrigin functions with high success rates. From Fig. 2 there are a lot of local peaks and valleys for this Rastrigin function, this problem is a little bit difficult than sphere function. However, EMPSO can solve the problem as well with few failures. In [9], the largest dimension is restricted to 200.

4.3 Example 3: Rosenbrock Function with n Variables and Difficulty Index of 2

$$f_3 = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2) \tag{5}$$

Where $-100 < x_i < 100$, $i = 1,2,....., n$, n is the dimensionality or number of variables to be optimized. The objective is to find the minimum of f_3 function and the related variable locations. Fig. 3 shows the surface plot of the function in 2 variables.

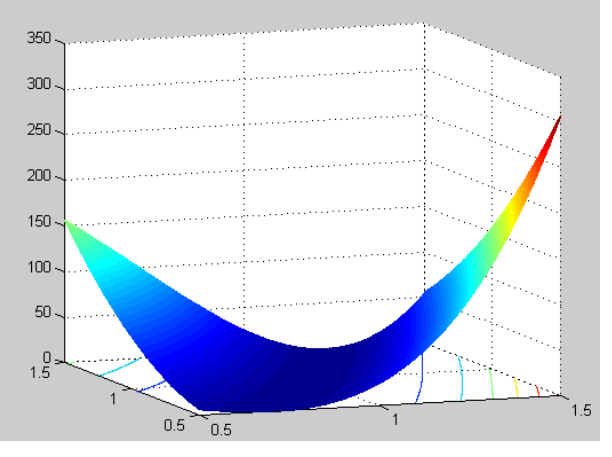


Fig. 3. Surface plot of the Rosenbrock function in n=2 variables

There is a unique minimum point with value of zero in the figure with location (1, 1). Population size is 600. Parameters for PSO and enhanced mechanism (EMPSO) are

(Algorithm #2)

(1). $c_1=c_2=1.9$, $\omega=1$, randxx=0.5 from the beginning.
(2). After one third of total generations, reset the parameters to new values such as $\omega=0.5$, randxx=0.99999.
(3). GA mutation rate is always fixed to 0.0035.
(4). One-variable enforced mutation is used.

From Table 3, the EMPSO shows its great capability to find the optimal solution with many generations for the high dimension Rosenbrock functions with almost full success rates. Because of the tight relation among all variables and sticked to 1, this problem will cost a lot of time as well as generations to finish finding the optimal solution. Dimension is up to 200 with many generations, so over 200, the generation number is expected exponentially increasing. Therefore, there still is progress to be studied in the future for dimension over 200. At present, the dimension is increased from 100 [9] to 200.

Table 3. Optimization results of Rosenbrock function

Number of variables (dimension)	Success rate	Generations to complete (average)	Mutate rate
2	50/50	47	0.0035
10	50/50	7749	0.0035
25	48/50	16879	0.0035
50	50/50	24641	0.0035
100	50/50	46217	0.0035
200	50/50	73333	0.0035

4.4 Example 4: Schafer Function with n Variables. Eq. (6) Shows n=2, (x, y) Only. The Difficulty Index is 3.

$$A1 = (\sin(\sqrt{x^2 + y^2}))^2 - 0.5$$
$$A2 = (1 + 0.001(x^2 + y^2))^2$$
$$f_4 = 0.5 - \frac{A1}{A2}. \tag{6}$$

Where $-100 < x_i < 100$, $i = 1,2,....,\ n$, n is the dimensionality or number of variables to be optimized. The objective is to find the maximum of f_4 function and the related variable locations. Fig. 4 shows the surface plot of the function in 2 variables. There is a unique maximum point with value of one in the figure with location (0, 0). Population size is 600. Parameters for PSO and enhanced mechanism (EMPSO) are

(Algorithm #3)

(1). c_1 is decreasing with generations from 2 to 1.5, c_2=3.8-c_1, ω=0.763, randxx=0.99999.
(2). GA mutation rate is increasing from 0.0035 to 0.01 with generation count.
(3). One-variable enforced mutation is used.

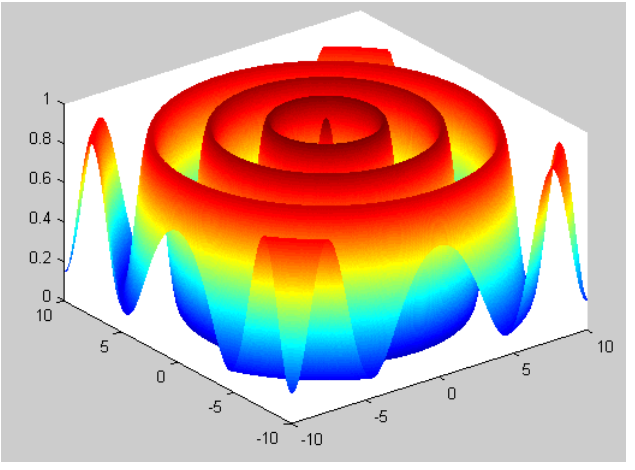


Fig. 4. Surface plot of the Schaffer function in n=2 variables

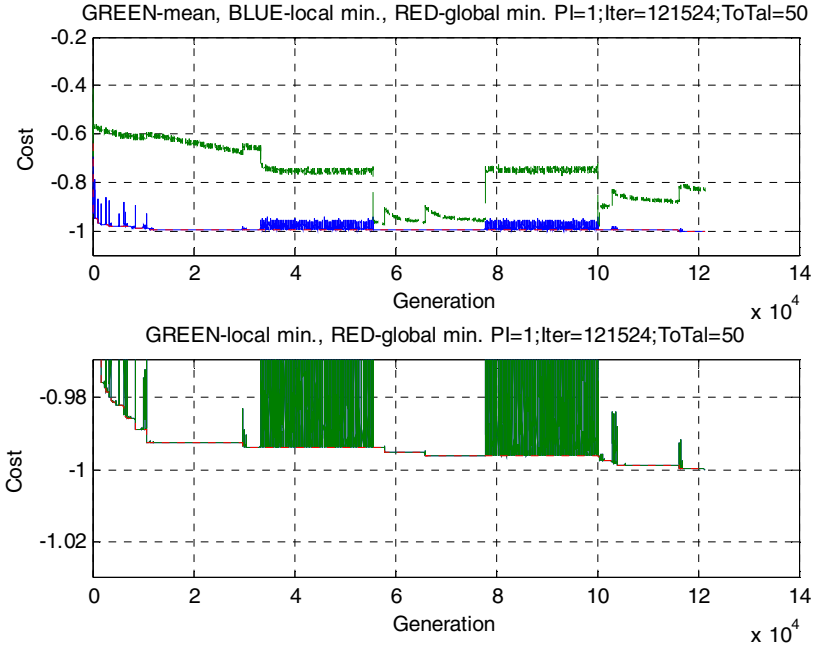


Fig. 5. One simulation result for Example 4.4

From Table 4, the EMPSO shows its great capability to find the optimal solution with many generations for the high dimension Schaffer functions with almost full success rates. From Fig. 4, if the algorithm is not efficient, then gebst is absorbed in the local traps region. It is relatively difficult to jump out of those traps in general. For

example, IMPSO, IPSO are used to overcome the trapping problem in low dimensionality problems. Furthermore the capability of jumping out of traps is increased by incorporating the one-variable enfored mutation mechanism and high GA mutation rate into EMPSO. Fig. 5 shows that after one-variable mutation's disturbance (blue regions in the lower part), global minimum (gbest) can move down gradually further. It takes minimization to do the whole performances in the PSO algorithm, so -1 is the true minimum or +1 is the true maximum in Fig. 5. Nearby information provided by the local maximum as shown in Fig. 4 is wrong, therefore, further modifications on particle/velocity is really difficult if original PSO algorithm (see eq.(1), (2)) is used. Therefore this Schaffer function has an index of difficulty of 3. The dimension in [9] is limited to 16 only, but 28 used in EMPSO.

Table 4. Optimization results of finding solutions of Schaffer functions

Number of variables (dimension)	Success rate	Generations to complete (average)	Mutate rate	Calling oneVar times
2	50/50	56	0.0035 to 0.01	3
4	50/50	128	0.0035 to 0.01	3
6	50/50	1745	0.0035 to 0.01	3
8	50/50	11427	0.0035 to 0.01	3
10	50/50	30528	0.0035 to 0.01	3
12	50/50	49195	0.0035 to 0.01	3
14	49/50	92415	0.0035 to 0.01	3
16	50/50	116328	0.0035 to 0.01	3
18	50/50	138385	0.0035 to 0.01	3
20	50/50	168559	0.0035 to 0.01	3
22	48/50	200465	0.0035 to 0.01	3
24	46/50	189222	0.0035 to 0.01	3
26	36/50	209565	0.0035 to 0.01	3
28	41/50	269290	0.0035 to 0.01	3

5 Conclusions

In this paper, an improved PSO with enforced mutation (EMPSO) has been proposed for handling four different problems with different difficulty indices. From the discussions above, one can find that the easy problems such as Sphere and Rastrigin functions with n-dimension can be easily overcome by adjusting PSO parameters in one step. For the intermediate difficulty of problems such as Rosenbrock, two steps of parameter setting are required in the PSO besides GA mutation. Lastly to the toughest problem such as Schaffer, the success to reach the final global optimum is relied highly on the enforced mutation mechanism when dimensionality of parameters is high. It is apparent from Fig. 5 that after the disturbing of one-variable enforced mutation, the final optimal solution can be reached by jumping out of the trap developed by local optimal locus.

The conclusion is that with optimization problems with different difficulty indices, firstly, the proposed EMPSO incorporates enforced mutation mechanism to the classical or standard PSO besides formal GA mutation; secondly, different settings of control parameters are required to set according to the relevant index of difficulty of the problem. The chances to find global solutions are increasing using this EMPSO algorithm to different problems. The dimension can be traced as high as 2000 in the first and the second examples. For the most difficult problem, the dimension is improved further from 16 to 28. With EMPSO, the possibility to find a global optimum is increased although the success rate is not 100% as shown in Table 4.

Acknowledgments. A financial support of DaYeh University for this study is highly acknowledged.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942–1948 (1995)
2. Engelbrecht, A.: Computational Intelligence, 2nd edn. John Wiley & Sons, West Sussex (2007)
3. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence, pp. 69–73 (1998)
4. Cai, X., Cui Z., Zeng, J., Tan, Y.: Individual Parameter Selection Strategy for Particle Swarm Optimization, Particle Swarm Optimization. Tech Education and Publishing (2009)
5. Vesterstrom, J., Thomsen, R.: A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. In: Proc. IEEE Congress Evolutionary Computation, Portland, OR, June 20-23, pp. 1980–1987 (2004)
6. Wagih, M., Elkamchouchi, H.: Application of Particle Swarm Algorithm in Smart Antenna Array Systems, Particle Swarm Optimization. Tech Education and Publishing (2009)
7. Amin, A., Hegazy, O.: Swarm Intelligence Applications in Electric Machines, Particle Swarm Optimization. Tech Education and Publishing (2009)
8. Chou, P.: Principles and Applications of Genetic Algorithms. ChiamHwa Book Company, Taipei (2006) (in Chinese)
9. Chou, P.: Improved Particle Swarm Optimization with Mutation. In: MSI 2009 IASTED International Conference, Beijing, China (2009)
10. Chou, P.: A Proposal for Improved Particle Swarm Intelligence. In: IEEE 3CA International Conference 2010, Tainan, Taiwan (2010)

Normalized Population Diversity in Particle Swarm Optimization*

Shi Cheng[1,2] and Yuhui Shi[2]

[1] Dept. of Electrical Engineering and Electronics,
University of Liverpool, Liverpool, UK
Shi.Cheng@liverpool.ac.uk
[2] Dept. of Electrical & Electronic Engineering,
Xi'an Jiaotong-Liverpool University, Suzhou, China
Yuhui.Shi@xjtlu.edu.cn

Abstract. Particle swarm optimization (PSO) algorithm can be viewed as a series of iterative matrix computation and its population diversity can be considered as an observation of the distribution of matrix elements. In this paper, PSO algorithm is first represented in the matrix format, then the PSO normalized population diversities are defined and discussed based on matrix analysis. Based on the analysis of the relationship between pairs of vectors in PSO solution matrix, different population diversities are defined for separable and non-separable problems, respectively. Experiments on benchmark functions are conducted and simulation results illustrate the effectiveness and usefulness of the proposed normalized population diversities.

Keywords: Particle swarm optimization, diversity, normalized population diversity, matrix analysis, vector norm, matrix norm, matrix iterative computation.

1 Introduction

Particle Swarm Optimization (PSO), which was introduced by Russ Eberhart and James Kennedy in 1995 [1,2], is a population-based evolutionary computation technique. It models the social behaviors observed in flocking birds. Each particle, which represents a potential solution, flies through the solution (search) space with a velocity that is dynamically adjusted according to its own and its companion's historical behaviors. The particles have a tendency to fly toward better search areas over the course of a search process [3].

Diversity has been defined to measure the search process of an evolutionary algorithm. Generally, it is not to measure whether the algorithm find a "good enough" solution or not, but to measure the distribution of individuals in the

* The authors' work was supported by National Natural Science Foundation of China under grant No. 60975080, and Suzhou Science and Technology Project under Grant No. SYJG0919.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 38–45, 2011.
© Springer-Verlag Berlin Heidelberg 2011

population (current solutions). Shi and Eberhart introduced three different definitions on population diversity to measure the PSO search process [4]. Olorunda and Engelbrecht utilized swarm diversity to measure the state of exploration or exploitation during particles searching [5]. Because different problems have different dynamic ranges, the dynamic ranges of these defined diversities generally will be different. As a consequence, the diversity observation on one problem will be different from that on another problem. Therefore it is necessary to have normalized diversity definitions.

In this paper, the basic PSO algorithm, fundamental concepts of matrix computation, and the importance of diversity are reviewed in Section 2. In Section 3, definitions of position diversity, velocity diversity, and cognitive diversity are given for separable problem and non-separable problem, respectively. In Section 4, experiments on measuring population diversity are tested on benchmark functions and simulation results are discussed to illustrate the effectiveness and usefulness of the proposed normalized diversity definitions. Finally, conclusions are given in Section 5 together with some remarks and future research directions.

2 Preliminaries

2.1 Particle Swarm Optimization

The original PSO algorithm is simple in concept and easy in implementation. The basic equations are usually given as follow [6]:

$$v_{ij} = wv_{ij} + c_1\text{rand}()(p_{ij} - x_{ij}) + c_2\text{Rand}()(p_{nj} - x_{ij}) \tag{1}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{2}$$

where x_{ij} represents a particle, i represents the number of particle which is from 1 to m, and j is for dimension from 1 to n.

The equations above can also be written in matrix form as follow:

$$\mathbf{V} = w\mathbf{V} + c_1\text{RAnd}()(\mathbf{P} - \mathbf{X}) + c_2\text{RAND}()(\mathbf{N} - \mathbf{X}) \tag{3}$$

$$\mathbf{X} = \mathbf{X} + \mathbf{V} \tag{4}$$

where RAnd() and RAND() are different for each matrix element, and

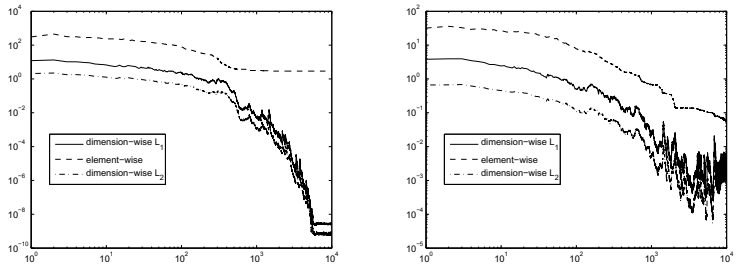
$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & x_{ij} & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & & v_{ij} & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} p_1 \\ p_2 \\ \cdots \\ p_m \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & & p_{ij} & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \quad \mathbf{N} = \begin{bmatrix} p_1^* \\ p_2^* \\ \cdots \\ p_m^* \end{bmatrix} = \begin{bmatrix} p_{11}^* & p_{12}^* & \cdots & p_{1n}^* \\ p_{21}^* & p_{22}^* & \cdots & p_{2n}^* \\ \vdots & & p_{ij}^* & \vdots \\ p_{m1}^* & p_{m2}^* & \cdots & p_{mn}^* \end{bmatrix}$$

The above four matrix are termed as position matrix $\mathbf{X}$, velocity matrix $\mathbf{V}$, cognitive (personal best) matrix $\mathbf{P}$, and social (neighboring best) matrix $\mathbf{N}$, which is a simplified personal matrix with only one particle's best position which is either the global best position in global star structure or a particle's personal best in this neighborhood in other structures, e.g., local ring.

Shi and Eberhart gave three definitions on population diversity, which are position diversity, velocity diversity and cognitive diversity [7]. According to the PSO matrix representation, diversity is a measurement of variance of different elements in each dimension or in whole matrix. Position diversity is used to measure the distribution of particles' current positions, that is, it concerns elements in matrix $\mathbf{X}$. Velocity diversity is used to measure the distribution of swarm's current velocity, that is, it concerns elements in matrix $\mathbf{V}$. Cognitive diversity measures the distribution of best positions for each particles find so far, that is, it concerns elements in matrix $\mathbf{P}$. Which diversity definition to be utilized to measure the diversity of swarm is determined by the property of particle swarm algorithms and the problems to be solved.

Each vector $\mathbf{x}_i$, where $\mathbf{x}_i = [x_{i1}, \cdots, x_{in}]$ in position matrix $\mathbf{X}$, is a solution of problem. An optimization problem can be a separable problem or a non-separable problem. For a separable problem, it is independent to evaluate the contribution of each x_{ij} to the fitness value of $\mathbf{x}_i$; while for non-separable problem, it is not independent. Therefore, for separable problems, it is preferred to use dimension-wise diversity measurement, while for non-separable problems it is preferred to use element-wise diversity measurement. This conclusion can be observed in Figure 1, which shows simulation results of different PSO population diversities for one separable problem (Fig. a) and one non-separable problem (Fig. b). The dimension-wise diversity measurement is better for separable function as shown in Fig. a, on the contrary, element-wise measurement is better for non-separable function as shown in Fig. b.



(a) (b)

Fig. 1. Different definition of PSO population diversity. Global star structure: (a) Separable Ackley's function: f_3 position, (b) Non-separable generalized Rosenbrock's function: f_4 position.

2.2 Vector Norm and Matrix Norm

A vector norm is a map function $f : \mathbb{R}^n \to \mathbb{R}$. All norms on $\mathbb{R}^n$ are equivalent, i.e., if $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are norms on $\mathbb{R}^n$, there exist positive constants, c_1 and c_2 such that $c_1\|\mathbf{x}\|_\alpha \leq \|\mathbf{x}\|_\beta \cdots \leq \mathbf{c_2}\|\mathbf{x}\|_\alpha$, Vector norm have the property that:

$$\|\mathbf{x}\|_\mathbf{1} \geq \|\mathbf{x}\|_\mathbf{2} \cdots \geq \|\mathbf{x}\|_\infty \tag{5}$$

Also, a matrix norm is a map function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$. The matrix norm have the properties that: $\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|$, and $\|A\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$.

By applying matrix norms in PSO, the meaning of matrix norm is as follows: for each dimension, calculating the sum of absolute position value for every particle, the maximum is the matrix norm L_1 for position matrix; for every particle, finding the sum of absolute position value in each dimension, the maximum is the matrix norm L_∞ for position matrix.

The distinction between matrix L_1 norm and matrix L_∞ norm is the perspectives taken on the position matrix. Matrix L_1 norm measures the largest value on dimension, while matrix L_∞ norm measures the largest value on particles.

Considered the property whether vectors are dependent on each other or not, vector norms are preferred to be applied to normalize population diversity for separable problems and matrix L_∞ norms are preferred to be used for non-separable problems.

3 Normalized Population Diversity

Vector norms are applied to normalize population diversity on separable problem, while matrix norms are applied to non-separable problem for the property that each dimension is not independent in non-separable problems. The matrix elements (position, velocity, personal best position) are normalized at first, and the definitions of PSO population diversities based on vector L_1 norm, which introduced by Cheng and Shi [8], are given as follow:

3.1 Position Diversity

Position diversity measures distribution of particles' current position. The swarm is going to diverge into wider search space or converge in a small area can be obtained from this measurement. Position diversity concerns the elements in position matrix.

Separable Problem. For separable problem, each vector in position matrix are independent. Vector norms are preferred to normalize the position, and three methods are as follow. These normalizations are based on the vector L_1 norm, or L_∞ norm, or maximum value of position: $x_{ij}^{nor} = x_{ij}/\|\mathbf{x}\|_1 = x_{ij}/\sum_{j=1}^{n} |x_{ij}|$,or $x_{ij}^{nor} = x_{ij}/\|\mathbf{x}\|_\infty = x_{ij}/\max|x_{ij}|$,or $x_{ij}^{nor} = x_{ij}/X_{\max}$. Considered the inequality (5), normalized position based on other vector norms is always larger than position based on L_1 norm and smaller than position based on L_∞ norm.

Normalized position diversities for separable problems are calculated as follow: $\bar{\mathbf{x}}^{nor} = \frac{1}{m}\sum_{i=1}^{m} x_{ij}^{nor}$, and $\mathbf{D}^p = \frac{1}{m}\sum_{i=1}^{m} |x_{ij}^{nor} - \bar{x}_j^{nor}|$, and $D^p = \frac{1}{n}\sum_{j=1}^{n} D_j^p$. where $\mathbf{D}^p = [D_1^p, \cdots, D_n^p]$ are the diversities on each dimension, and D^p is the normalized position diversity for particles.

Non-separable Problem. For non-separable problem, a vector in position matrix is relative to other vector or vectors. This connection should be considered in diversity measurement. Three methods are preferred to normalize the position of each particle $\max|x_{ij}|$ in matrix $\mathbf{X}$, matrix L_∞ norm for $\mathbf{X}$, or the maximum value of position. Normalized position is as follows: $x_{ij}^{nor} = x_{ij}/\max|x_{ij}|$, or $x_{ij}^{nor} = x_{ij}/\|\mathbf{X}\|_\infty = x_{ij}/\max_{1\le i\le m}\sum_{j=1}^{n}|x_{ij}|$, or $x_{ij}^{nor} = x_{ij}/X_{\max}$.

After normalized the position, normalized position diversity for non-separable problems is calculated as follows: $\bar{x}^{nor} = \frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^{nor}$, and $D^p = \frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n}|x_{ij}^{nor} - \bar{x}^{nor}|$. where D^p is the normalized position diversity for particles at this running step.

3.2 Velocity Diversity

Velocity diversity, which gives the tendency information of particles, measures the distribution of particles' current velocity. In other words, velocity diversity measures the "activity" information of particles. Based on the measurement of velocity diversity, particle's tendency of expansion or convergence could be obtained.

Separable Problem. Vector in velocity matrix is independent for separable problem. Vector norm L_1, or L_∞, or maximum value of velocity is applied to normalize velocity: $v_{ij}^{nor} = v_{ij}/\|\mathbf{v}\|_1 = v_{ij}/\sum_{j=1}^{n}|v_{ij}|$, or $v_{ij}^{nor} = v_{ij}/\|\mathbf{v}\|_\infty = v_{ij}/\max|v_{ij}|$, or $v_{ij}^{nor} = v_{ij}/V_{\max}$.

Normalized velocity diversities for separable problems are calculated as follow: $\bar{\mathbf{v}}^{nor} = \frac{1}{m}\sum_{i=1}^{m} v_{ij}^{nor}$, and $\mathbf{D}^v = \frac{1}{m}\sum_{i=1}^{m}|v_{ij}^{nor} - \bar{v}_j^{nor}|$, and $D^v = \frac{1}{n}\sum_{j=1}^{n} D_j^v$. where $\mathbf{D}^v = [D_1^v, \cdots, D_n^v]$ are the diversities on each dimension, and D^v is the normalized velocity diversity for particles.

Non-separable Problem. For non-separable problem, vectors is not independent in velocity matrix. Three operators: $\max|v_{ij}|$ in velocity matrix, or matrix L_∞ norm of $\mathbf{V}$, or maximum value of velocity is applied to normalize the velocity. $v_{ij}^{nor} = v_{ij}/\max|v_{ij}|$, or $v_{ij}^{nor} = v_{ij}/\|\mathbf{V}\|_\infty = v_{ij}/\max_{1\le i\le m}\sum_{j=1}^{n}|v_{ij}|$, or $v_{ij}^{nor} = v_{ij}/V_{\max}$.

Normalized velocity diversity for non-separable problems is calculated as follows: $\bar{v}^{nor} = \frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^{nor}$, and $D^v = \frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n}|v_{ij}^{nor} - \bar{v}^{nor}|$. where D^v is the normalized velocity diversity for particles at this running step.

3.3 Cognitive Diversity

Cognitive diversity represents the target distribution of all particles found currently. The measurement of cognitive diversity is as same as position diversity except using each particle's current personal best position instead of current position. Therefore, the analysis for position diversity is also effective for cognitive diversity.

Separable Problem. The normalized cognitive positions are as follow: $p_{ij}^{nor} = p_{ij}/\|\mathbf{p}\|_1 = p_{ij}/\sum_{j=1}^{n}|p_{ij}|$, or $p_{ij}^{nor} = p_{ij}/\|\mathbf{p}\|_\infty = p_{ij}/\max|p_{ij}|$, or $p_{ij}^{nor} = p_{ij}/X_{\max}$.

Normalized cognitive diversities for separable problems are calculated as follow: $\bar{\mathbf{p}}^{nor} = \frac{1}{m}\sum_{i=1}^{m} v_{ij}^{nor}$, and $\mathbf{D}^c = \frac{1}{m}\sum_{i=1}^{m}|v_{ij}^{nor} - \bar{v}_j^{nor}|$, and $D^c = \frac{1}{n}\sum_{j=1}^{n} D_j^v$. where $\mathbf{D}^c = [D_1^c, \cdots, D_n^c]$ are the diversities on each dimension, and D^c is the normalized cognitive diversity for particles.

Non-separable Problem. Like the definition of position diversity, the normalized personal best positions are as follow: $p_{ij}^{nor} = p_{ij}/\max_{i,j}|p_{ij}|$, or $p_{ij}^{nor} = p_{ij}/\|\mathbf{P}\|_\infty = p_{ij}/\max_{1\le i\le m}\sum_{j=1}^{n}|p_{ij}|$, or $p_{ij}^{nor} = p_{ij}/X_{\max}$

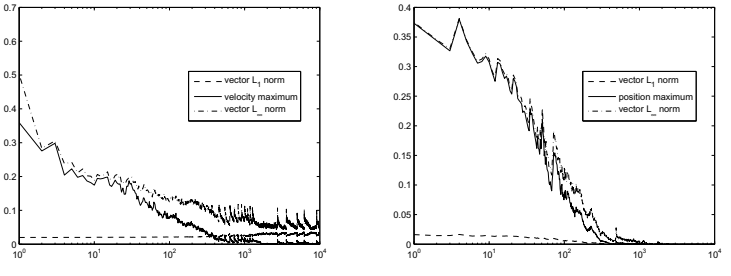
Normalized cognitive diversity for non-separable problems is calculated as follows: $\bar{p}^{nor} = \frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^{nor}$, and $D^c = \frac{1}{m\times n}\sum_{i=1}^{m}\sum_{j=1}^{n}|p_{ij}^{nor} - \bar{p}^{nor}|$. where D^c is the normalized cognitive diversity for particles at this iterative step.

4 Experimental Studies

The experiments have been conducted to test more than 10 benchmark functions; some of them are given in the Table 1. All functions are tested for 50 times, and random shift of the location of optimum is utilized in dimensions each time. For the reason of generalization, whether each benchmark function is either a unimodal or a multimodal function as well as a separable or a non-separable function is known.

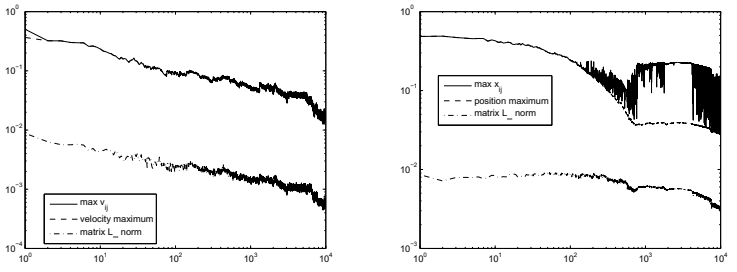
Figure 2 gives a diversity measurement for separable problems, which (a) is velocity diversity of unimodal function f_1 and (b) is position diversity of multimodal function f_3. Fig. a and b showed that diversity based on L_1 norm nearly have the same curve as diversity based on the maximum value of the position matrix.

Figure 3 gives a diversity measurement for non-separable problems, which (a) is velocity diversity of unimodal function f_2 and (b) is position diversity of



(a) (b)

Fig. 2. Normalized PSO population diversity for separable problem. Local ring structure: (a) unimodal function f_1 velocity diversity; Global star structure: (b) multimodal function f_3 position diversity.



(a) (b)

Fig. 3. Normalized PSO population diversity for non-separable problem. Local ring structure: (a) unimodal function f_2 velocity diversity; Global star structure: (b) multimodal function f_4 position diversity.

Table 1. Representative functions used in our experimental study, where n is the dimension of the function, $\mathbf{z} = (\mathbf{x} - \mathbf{o})$, o_i is an randomly generated number in each function's search space S, global optimum $\mathbf{x}^* = \mathbf{o}$, $f_{\min}$ is the minimum value of the function, and $S \subseteq R^n$

Function name	Test function	n	S	$f_{\min}$
Quadric Noise	$f_1(\mathbf{x}) = \sum_{i=1}^{n} iz_i^4 + \text{random}[0,1)$	100	$[-1.28, 1.28]^n$	bias[4]
Schwefel's p1.2	$f_2(\mathbf{x}) = \sum_{i=1}^{n} (\sum_{k=1}^{i} z_k)^2$	100	$[-100, 100]^n$	bias[2]
Ackley	$f_3(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)} + 20 + e$	100	$[-32, 32]^n$	bias[9]
Generalized Rosenbrock	$f_4(\mathbf{x}) = \sum_{i=1}^{n} [100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2]$	100	$[-10, 10]^n$	bias[5]

multimodal function f_4. Fig.a and b showed that diversity based on max x_{ij} nearly have the same curve as diversity based on the maximum value of the position matrix.

As Figures shown, this definition of normalized diversity measurement gives some useful information during particles search process. Position diversity and Velocity diversity always have a continuous vibrate, this proved that particles "fly" from one side of optimum to another side on each dimension continually.

5 Conclusion

This paper proposed an analysis of population diversity based on the category of separable and non-separable problems. If vectors are not independent in position matrix; diversity observation should consider the correlation among dependent vectors. Considered the property whether vectors are dependent on each other or not, vector norms are preferred to normalize population diversity for separable problems and matrix L_∞ norm is preferred to be used for non-separable problems.

Particles on the state of "expansion" or "converge" can be determined by this diversity measurement. After obtained this information, performance of optimization algorithm can be improved by adjusting population diversity dynamically during PSO search process. Particles with different topology structure also have different vector dependence in position or velocity matrix. Seeking the influence of topology structure and vector partial dependence analysis is the research need to be explored further.

The idea of normalized population diversity measuring can also be applied to other evolutionary algorithms, e.g., genetic algorithm, differential evolution for evolutionary algorithms have the same concepts of current population solutions and search step. The performance of evolutionary algorithms can be improved on this measurement of population diversity. Dynamically adjusting the population diversity controls the algorithm's ability of exploration or exploitation, hence, algorithm has large possibility to reach optimum.

References

1. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Processings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Processings of IEEE International Conference on Neural Networks (ICNN), pp. 1942–1948 (1995)
3. Eberhart, R., Shi, Y.: Computational Intelligence: Concepts to Implementations, 1st edn. Morgan Kaufmann Publisher, San Francisco (2007)
4. Shi, Y., Eberhart, R.C.: Population diversity of particle swarms. In: Proceedings of the 2008 Congress on Evolutionary Computation, pp. 1063–1067 (2008)
5. Olorunda, O., Engelbrecht, A.P.: Measuring exploration/exploitation in particle swarms using swarm diversity. In: Proceedings of the 2008 Congress on Evolutionary Computation, pp. 1128–1134 (2008)
6. Eberhart, R., Shi, Y.: Particle swarm optimization: Developments, applications and resources. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 81–86 (2001)
7. Shi, Y., Eberhart, R.: Monitoring of particle swarm optimization. Frontiers of Computer Science 3(1), 31–37 (2009)
8. Cheng, S., Shi, Y.: Diversity control in particle swarm optimization. In: Proceedings of IEEE Swarm Intelligence Symposium 2011, Paris, France (April 2011)

Particle Swarm Optimization with Disagreements

Andrei Lihu and Ştefan Holban

Department of Computer Science, Politehnica University of Timişoara,
Bd. Vasile Pârvan, 300223 Timişoara, Romania
andrei.lihu@gmail.com,stefan@cs.upt.ro

Abstract. This paper introduces an enhancement to the particle swarm optimization algorithms that models a characteristic of social groups: the disagreements between individuals. After a short introduction, we describe the new concept theoretically and define a special type of particle swarm optimization with disagreements: the 6σ-PSOD. Based on it, we conduct some tests proving that it can perform better, having strengthened neighborhood focus using partial disagreements and enhanced exploration capabilities through extreme disagreements.

Keywords: disagreements, partial, extreme, 6σ-PSOD, particle swarm optimization, standard deviation.

1 Introduction

The particle swarm optimization (PSO) is a population based optimization algorithm, initially described by Eberhart and Kennedy in [1]. It implements the social mind metaphor, simulating the social behavior of different groups from the animal kingdom, such as birds or fish.

One of the main advantages of PSO over other optimization techniques is the ability to operate without the need of gradient information. It successfully solves nonlinear and multi-objective problems from various fields of human activity and at the time of this writing, it is a top competitor among optimization algorithms.

Briefly, PSO is better described as a swarm of particles flying iteratively through the hyperspace of solutions until a termination condition is met. Every particle has a velocity and a position, knows his neighborhood's best position and its personal one; it can check whether a better position is found by using a fitness function. Each iteration, a particle updates its velocity and position based on its own experience — *the cognitive component* of the algorithm's updating principle, and based on its neighborhood experience — *the social component.*

Important improvements to the original algorithm consist in ameliorating convergence and increasing diversity inside the swarm, as stated in [2]. While improving convergence gets the job done by discovering faster the solution, increasing the diversity helps not getting trapped into local minima. Finding a way that can accommodate both of these objectives is hard, because they are somehow opposite, therefore compromises must be made. We approached this

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 46–55, 2011.
© Springer-Verlag Berlin Heidelberg 2011

problem by completing the social background of PSO with a new behavior: the disagreements.

We thought that a particle in PSO can oppose its group's way by having a modified opinion, therefore a particle can follow mildly or extremely different paths from the regular one, with a given probability.

After testing and comparing our results with popular PSO variants (using a comprehensive evolutionary framework — Java EvA2 [3]), we concluded that our new approach has good results. It improves convergence by having a better focus inside the neighborhood through partial disagreements, and increases the diversity when extreme disagreements are generated, making it suitable for solving plateau and/or multi-modal functions.

2 Standard Particle Swarm Optimization

Following Berg's notation in [2], PSO can be described as follows: let n be the dimension of the solution hyperspace H^n, let s be the number of particles from the swarm, and let i be the index of a particle, such that $i \in \overline{1 \dots s}$. Each particle i has the following variables: x_i — the current position, v_i — the current velocity, y_i — the current best position, $\hat{y}$ — neighborhood's best. We consider the function f to be minimized. The PSO consists of three phases: initialization, iterations, termination.

Initially, the particles' positions are set uniformly in the search space. Then, best positions are updated using (3) and (4).

In the iterations' phase, both velocities and positions are updated using (1) and (2):

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \ , \quad (1)$$

where c_1 is the personal coefficient, c_2 is the social coefficient, $c_1, c_2 \in (0, 2]$. r_1 and r_2 are random vectors, such that: $r_1, r_2 \sim U(0, 1)$. The first term of (1) is the previous velocity influenced by an inertial weight w. The second term is the personal component that makes the particle move toward its best personal position so far, and the third term makes the particle to turn to neighborhood's best position found so far:

$$x_i(t+1) = x_i(t) + v_i(t+1) \ . \quad (2)$$

At each iteration, y_i and $\hat{y}$ are updated using the formulae:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases} \ . \quad (3)$$

$$\hat{y}(t) \in \{y_0(t), y_1(t), \dots, y_s(t) | f(\hat{y}(t))\} = \\ min\{f(y_0(t)), f(y_1(t)), \dots, f(y_s(t))\} \ . \quad (4)$$

Termination occurs when a given criterion is met (a number of fitness calls or iterations, stagnation, etc.).

3 Preliminary Studies

In the original PSO updating principle (1), the inertial weight w did not exist. Shi and Eberhart introduced w in [4] and it was one of the first improvements towards a good convergence of PSO. The issue that it tried to solve was the "swarm explosion" phenomenon and the lack of convergence. Another classical improvement towards increasing convergence was the introduction of the constriction coefficient by Clerc and Kennedy in [5]. They showed that the particles have an oscillatory movement and they elaborated a constriction-based PSO with the standard configuration: $\chi = 0.729$ and $c_1 = c_2 = 2.05$. We can also mention here Bergh's guaranteed convergence PSO (GCPSO) as in [2].

The other main direction in enhancements consisted in improving diversity, to not get trapped in local minima, thus avoiding premature convergence. In this regard, first came the introduction of the local neighborhoods — the *lbest* variant of PSO with topologies. Recent approaches include a dissipative version of PSO as in [6].

One of the variants that try to solve both the problem of convergence and that of diversity is Chen and Li's PSO in [7], [8] and [9] with guaranteed convergence and controllable random exploration velocity (PSO-CREV). The idea presented in this paper has the same aim.

4 Disagreements

Disagreements are a social phenomenon that leads to diversity and heightened awareness of current problems. Therefore, they can be applied in any social algorithm like PSO to increase the diversity of solutions and to strengthen the search focus in local neighborhood, resulting in better convergence. Replacing the first term in (1) with a generic $C_v(t)$, denoting the velocity component, the second term in (3) with $C_c(t, x_i, y_i)$ denoting the cognitive component, the third term with $C_s(t, x_i, \hat{y})$ and making the substitution in (2) where we also change the position component with a generic one, C_x, we obtain the generalized updating equation:

$$\begin{gathered} x_i(t+1) = C_{\mathrm{x}}(t, x_i) + C_{\mathrm{v}}(t) + C_{\mathrm{c}}(t, x_i, y_i) + C_{\mathrm{s}}(t, x_i, \hat{y}) + \zeta \ , \\ C_{\mathrm{c}}(t, x_i, y_i) \rightarrow y_i, C_{\mathrm{s}}(t, x_i, \hat{y}) \rightarrow \hat{y} \ . \end{gathered} \tag{5}$$

$C_c(t, x_i, y_i) \rightarrow y_i$ should be read as "the result of C_c tends to y_i" and $C_s(t, x_i, \hat{y}) \rightarrow \hat{y}$ should be read as "the result of C_s tends to $\hat{y}$". ζ is usually 0 and can accommodate any other more elaborate variant of PSO that may consist of other components too.

A disagreement is defined as a function that takes values in the hyperspace of the results of the social component C_s, which is H^n. One can define the disagreements as the family of D functions for which the following property holds true:

$$F_{\mathrm{D}} = \{D : H^n \rightarrow H^n | D(z) \neq z\}, \ \forall z \in H^n \ . \tag{6}$$

Let P_{all} be the set of all types of PSO that contain the social component C_{s} in the updating principle,

$$P_{\text{all}} = \{PSO | PSO \text{ has } C_{\text{s}} \text{ in its updating principle}\} \ , \tag{7}$$

and let Δ_v be a subset from all possible subsets, Δ_{all}:

$$\Delta_v \subset \{\emptyset_{\text{D}}, F_{\text{D}}\} , \ \ \Delta_v \in \Delta_{\text{all}} \ , \tag{8}$$

where $\emptyset_{\text{D}}$ is a no-op (no disagreement). We define the "disagreement injector" on any social PSO as:

$$\Psi_{\text{PSO}} : \{P_{\text{all}}, \Delta_{\text{all}}, \rho_{\text{all}}\} \rightarrow P_{\text{all}}, \ \ \Psi_{PSO}(P_i, \Delta_v, \rho) = P_{i\text{D}}, \ \ P_i, P_{i\text{D}} \in P_{\text{all}} \ , \tag{9}$$

where P_i is one of the many PSO variants with social component, and $P_{i\text{D}}$ is the resulting particle swarm optimization with disagreements. ρ is *a decision function* called "disagreement selector" that takes a set of disagreements (Δ_v) as argument at iteration t (from all iterations t_{all}) for a particle i and decides which disagreement is invoked:

$$\rho : \{\Delta_{\text{all}} \times t_{\text{all}} \times s\} \rightarrow \Delta_{\text{all}}, \ \ \rho(\Delta_v, t, i) = D_i, \ \ D_i \in \Delta_v \ . \tag{10}$$

After we apply the injection operator Ψ_{PSO}, the updating principle of the newly obtained PSO, now called PSOD (particle swarm optimization with disagreements) is transformed from (5) to:

$$\begin{aligned} x_i(t+1) = C_{\text{x}}(t, x_i) + C_{\text{v}}(t) + C_{\text{c}}(t, x_i, y_i) + D_i(C_{\text{s}}(t, x_i, \hat{y})) + \zeta \ , \\ C_{\text{c}}(t, x_i, y_i) \rightarrow y_i, C_{\text{s}}(t, x_i, \hat{y}) \rightarrow D_i(\hat{y}) \ . \end{aligned} \tag{11}$$

In this way we introduced the concept of disagreements as a special operator that can be applied to any social PSO without modifying the internals of the algorithm. The social component can vary in implementation from algorithm to algorithm, but the injection operator can be applied in any case.

Disagreement operators should not be confused with mutation of any kind. A disagreement operator can affect only the social component of the PSO and, as described above, it has a more elaborate structure and function.

In the following section we will define a practical approach to disagreements in PSO: a PSO disagreement operator called "the 6σ-PSOD operator".

5 6σ-PSOD Operator

The 6σ-PSOD operator, $\Psi_{6\sigma-\text{PSOD}}$, is defined in terms of (9) as:

$$\Psi_{6\sigma-\text{PSOD}}(P_i) = \Psi_{PSO}(P_i, \Delta_{6\sigma}, \rho_{6\sigma}) \ . \tag{12}$$

In this case, the subset of disagreements (the Δ_v that contains the disagreements, $D_{6\sigma i}$) is defined as:

$$\Delta_{6\sigma} = \{\emptyset_{\text{D}}, D_{6\sigma a}, D_{6\sigma b}\} \ . \tag{13}$$

The first type of disagreement employed here is $D_{6\sigma a}$, which is a partial disagreement because it is a function that multiplies member-wise (a Hadamard product) the social component C_s by a vector r, which has its components uniformly distributed in the interval $[-1, +1]$.

$$D_{6\sigma a}(z) = r_p \otimes z, \;\; r_p \sim U(-1, +1), \;\; z \in H^n, p \in \overline{1 \ldots |r|} \;\; . \tag{14}$$

The second type of disagreement is the extreme disagreement $D_{6\sigma b}$, which multiplies member-wise the social component by a vector r containing random uniformly distributed values in the intervals $[-2, -1]$ and $[+1, +2]$:

$$D_{6\sigma b}(z) = r_e \otimes z, \;\; r_e = r_p + sgn(r_p), \;\; r_p \sim U(-1, +1), \\ z \in H^n, p \in \overline{1 \ldots |r|} \;\; . \tag{15}$$

A rudimentary visualization of these concepts can be seen in Fig. 1:

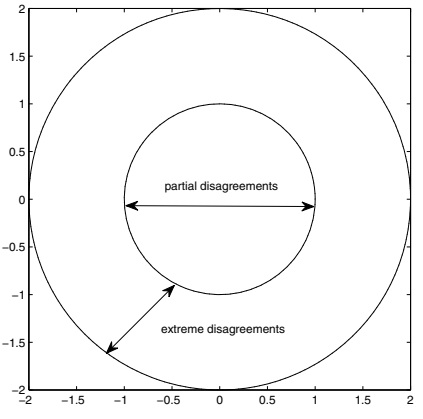


Fig. 1. Distribution of disagreement types in concentric circles. Two areas are shown here: for partial disagreements — r_p values fall inside the the inner circle, while for extreme disagreements — r_e values fall inbetween the inner and the outer circle.

At each iteration t, for each particle i, we generate $\theta(t, i) \sim \mathcal{N}(\mu_{6\sigma}, \sigma_{6\sigma}^2)$ and $\theta_1(t, i) \sim \mathcal{N}(\mu_{6\sigma}, \sigma_1^2)$, $\sigma_{6\sigma} \geq \sigma_1$. We define the following Gaussian regions for θ:

1. The first region accounts for approx. 68.2 % of the bell curve (first 2 σs) and it is defined as:
$$R_{1,2\sigma} = (\mu_{6\sigma} - \sigma_{6\sigma}) \cup (\mu_{6\sigma} + \sigma_{6\sigma}) \;\; . \tag{16}$$
2. The second region accounts for approx. 27.2 % of the bell curve (next 2 σs) and it is defined as:
$$R_{3,4\sigma} = (\mu_{6\sigma} - 2\,\sigma_{6\sigma}, \mu_{6\sigma} - \sigma_{6\sigma}] \cup [\mu_{6\sigma} + \sigma_{6\sigma}, \mu_{6\sigma} + 2\,\sigma_{6\sigma}) \;\; . \tag{17}$$

3. The third region accounts for approx. 4.6 % of the bell curve (next 2 σs and the rest of what remains under the graphic of the Gaussian function) and it is defined as:

$$R_{5,6\sigma} = (-\infty, \mu_{6\sigma} - 2\,\sigma_{6\sigma}] \cup [\,\mu_{6\sigma} + 2\,\sigma_{6\sigma}), +\infty) \ . \tag{18}$$

Based on (13), (14), (15) and the above defined Gaussian regions, the selector function is defined as:

$$\rho_{6\sigma}(\Delta_{6\sigma}, t, i) = \begin{cases} \emptyset_{\mathrm{D}} & \text{if } \theta_1(t,i) \in R_{1,2\sigma} \\ D_{6\sigma a} & \text{if } \theta_1(t,i) \in R_{3,4\sigma} \\ D_{6\sigma b} & \text{if } \theta_1(t,i) \in R_{5,6\sigma} \end{cases} \ . \tag{19}$$

The updating principle from (11) is transformed into:

$$\begin{gathered} x_i(t+1) = C_{\mathrm{x}}(t, x_i) + C_{\mathrm{v}}(t) + C_{\mathrm{c}}(t, x_i, y_i) + D_{6\sigma i}(C_{\mathrm{s}}(t, x_i, \hat{y})) + \zeta \ , \\ C_{\mathrm{c}}(t, x_i, y_i) \to y_i, C_{\mathrm{s}}(t, x_i, \hat{y}) \to D_{6\sigma i}(\hat{y}) \ . \end{gathered} \tag{20}$$

6 Experimental Setup

In order to test the effect of the 6σ-PSOD operator on the PSO performance, we have chosen a standard PSO with constriction as in [5], called SPSO, with the following configuration: $\chi = 0.729$ and $c_1 = c_2 = 2.05$. The second chosen algorithm was Pedersen's PSO-VG from [10], a social-only inertial PSO lacking the cognitive component, with the following configuration: $w = 0.729$ and $c_2 = 1.494$.

We transformed these two algorithms into their disagreement-enabled counterparts: $\Psi_{6\sigma-\mathrm{PSOD}}(SPSO) = SPSOD_{6\sigma}$ and $\Psi_{6\sigma-\mathrm{PSOD}}(PSO-VG) = PSO-VGD_{6\sigma}$, with $\sigma_{6\sigma} = 1$, $\sigma_1 = 0.7$ and $\mu_{6\sigma} = 0$.

Both algorithms used the grid topology. For each, we measured the mean fitness value and its standard deviation across 25 runs, for 25 and 50 particles in swarm, on 4 popular benchmark problems. We repeated this for a hyperspace of 10 and 30 dimensions. Algorithms terminated after 50 000 fitness evaluations.

The test problems we used were: Generalized Rosenbrock (F_2), Shifted Rastringin (F_{14}), Shifted Schwefel (F_{21}) and Griewank (F_5).

1. We started with Generalized Rosenbrock to see how PSODs behave on plateau functions:

$$F_2(X) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2), \quad X \in \mathbb{R}^n \ . \tag{21}$$

2. Then, we used Shifted Rastringin to check for behavior of convergence:

$$F_{14}(X) = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10), \quad X \in [-5, +5]^n, \quad Z = X - o \ . \tag{22}$$

3. With Shifted Schwefel we tested for robustness:

$$F_{21}(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j \right)^2, \quad X \in [-100, +100]^n, \quad Z = X - o \ . \tag{23}$$

4. Finally, with Griewank we checked if the new behavior helps escaping local minima:

$$F_5(X) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad X \in [-600, 600]^n \ . \tag{24}$$

Java EvA2 ([3]) was used to test the algorithms and implement the above described PSODs, because we considered it the most advanced tool for evolutionary research at the moment. The 6σ-PSOD operator corresponds to the 'SixSigma' variant of the PSODs we have developed in our modified Java EvA2 library. Sources are currently available at `https://github.com/andrei-lihu/Eva2-AL`.

7 Results

Tables 1–4 show the results of our experiments:

Table 1. Benchmark results for $n = 10$ dimensions (F_2 and F_{14})

		$\mathbf{F_2}$		$\mathbf{F_{14}}$	
Algorithm	**s**	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
SPSO	25	1.394	1.838	8.789	5.310
	50	4.142	13.959	4.815	2.543
$\text{SPSOD}_{6\sigma}$	25	1.020	1.560	8.198	5.311
	50	1.150	0.776	5.651	3.053
PSO-VG	25	0.326	1.081	26.317	14.601
	50	0.488	1.299	17.838	14.685
$\text{PSO} - \text{VGD}_{6\sigma}$	25	0.638	1.461	24.035	13.630
	50	0.168	0.780	14.685	8.316

The convergence graph from Fig. 2 shows how the 6σ-PSOD operator improves convergence by having a stronger neighborhood focus and how escapes local minima in a highly multi-modal and disturbed environment.

Fig. 3 illustrates how PSODs tackle plateau functions better than their PSOD counterparts. The possibility to extremely disagree helps them make the switch to better fitness values, as it can be noticed in the median part of the graphic.

All results show without any doubt that there is either a marginal improvement or a major one. Through partial disagreements the exploitation is increased and through extreme disagreements exploration gets a new twist. This proves empirically that adding disagreements can do no harm to any PSO, yet it can only improve it.

Table 2. Benchmark results for $n = 10$ dimensions (F_{21} and F_5)

		$\mathbf{F_{21}}$		$\mathbf{F_5}$	
Algorithm	**s**	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
SPSO	25	5.654×10^{-29}	$9.823 \times 10^{-}29$	8.475×10^{-42}	1.758×10^{-41}
	50	3.858×10^{-22}	8.718×10^{-22}	4.025×10^{-22}	1.043×10^{-21}
$\text{SPSOD}_{6\sigma}$	25	6.462×10^{-29}	9.634×10^{-29}	8.784×10^{-41}	2.070×10^{-40}
	50	2.348×10^{-22}	4.057×10^{-22}	2.929×10^{-22}	7.038×10^{-22}
PSO-VG	25	2.322×10^{-28}	2.019×10^{-28}	2.542×10^{-73}	1.248×10^{-72}
	50	3.811×10^{-29}	7.854×10^{-29}	5.756×10^{-29}	1.662×10^{-28}
$\text{PSO} - \text{VGD}_{6\sigma}$	25	1.777×10^{-28}	1.696×10^{-28}	1.733×10^{-73}	7.563×10^{-73}
	50	2.894×10^{-29}	1.211×10^{-29}	1.104×10^{-28}	2.775×10^{-28}

Table 3. Benchmark results for $n = 30$ dimensions (F_2 and F_{14})

		$\mathbf{F_2}$		$\mathbf{F_{14}}$	
Algorithm	**s**	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
SPSO	25	34.883	26.412	78.644	19.384
	50	48.664	35.162	61.737	13.837
$\text{SPSOD}_{6\sigma}$	25	35.597	47.031	76.053	21.463
	50	42.036	29.311	58.556	17.989
PSO-VG	25	39.156	25.735	146.764	51.717
	50	46.586	38.982	101.563	22.511
$\text{PSO} - \text{VGD}_{6\sigma}$	25	36.424	25.265	146.289	36.213
	50	43.097	27.72	100.086	23.657

Table 4. Benchmark results for $n = 30$ dimensions (F_{21} and F_5)

		$\mathbf{F_{21}}$		$\mathbf{F_5}$	
Algorithm	**s**	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
SPSO	25	1012.220	3371.174	258.254	697.851
	50	426.382	1166.260	125.508	436.200
$\text{SPSOD}_{6\sigma}$	25	291.037	1102.573	174.206	581.953
	50	369.314	902.364	121.468	418.583
PSO-VG	25	6391.002	5902.578	2052.589	2171.947
	50	6090.143	6295.384	2838.752	2483.361
$\text{PSO} - \text{VGD}_{6\sigma}$	25	5774.600	5308.191	2029.044	1656.979
	50	3820.831	4855.517	1891.053	2478.797

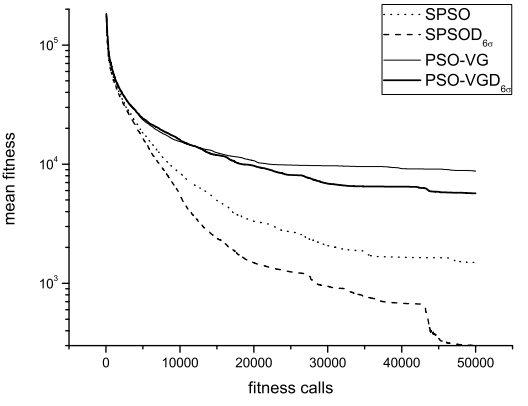


Fig. 2. Convergence graph for F_{21} (30 dimensions; $s = 25$). Comparing in pairs, it is obvious that PSOD variants have better convergence than their original PSO counterparts.

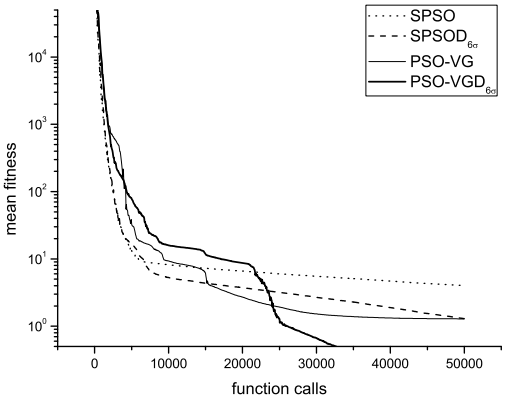


Fig. 3. Multi-run convergence graph for F_2 (10 dimensions; $s = 50$). In the median part of the graph PSODs make a good turn toward a better convergence and a better final solution.

8 Conclusion

We introduced a new metaphor in PSO that models disagreements between social groups, and starting from it we implemented and tested a disagreement operator that is based on the normal distribution: the 6σ-PSOD operator.

Our tests have shown that Particle Swarm Optimization with Disagreements, under the considered form of the 6σ-PSOD operator, is a beneficial enhancement

to any PSO because it can improve results when solving multi-modal and/or difficult plateau functions.

This paper only introduces the new idea with preliminary results. Further work is required to develop, analyze and test new PSOD operators.

Acknowledgments. This work was developed in the frame of PNII-IDEI-PCE-ID923-2009 CNCSIS – UEFISCSU grant and was partially supported by the strategic grant POSDRU 6/1.5/S/13-2008 of the Ministry of Labor, Family and Social Protection, Romania, co-financed by the European Social Fund – Investing in People.

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
2. Bergh, F.: An Analysis of Particle Swarm Optimizers (PhD thesis). University of Pretoria, Pretoria (2001)
3. EvA2 Project Homepage, http://www.ra.cs.uni-tuebingen.de/software/EvA2 (accessed in January 2011)
4. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: IEEE International Conference of Evolutionary Computation (1998)
5. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
6. Xie, X.-F., Zhang, W.-J., Yang, Z.-L.: Dissipative particle swarm optimization. In: Proceedings of the Congress of Evolutionary Computation, Honolulu, HI, USA, pp. 1456–1461 (2002)
7. Chen, X., Li, Y.: A Modified PSO Structure Resulting in High Exploration Ability With Convergence Guaranteed. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37(5), 1271–1289 (2007)
8. Chen, X., Li, Y.: Enhance Computational Efficiency of Neural Network Predictive Control Using PSO with Controllable Random Exploration Velocity. In: Liu, D., Fei, S., Hou, Z.-G., Zhang, H., Sun, C. (eds.) ISNN 2007. LNCS, vol. 4491, pp. 813–823. Springer, Heidelberg (2007)
9. Chen, X., Li, Y.: On Convergence and Parameters Selection of an Improved Particle Swarm Optimization. International Journal of Control, Automation, and Systems 6(4), 559–570 (2008)
10. Pedersen, M.E.H., Chipperfield, A.J.: Simplifying particle swarm optimization. Applied Soft Computing 10, 618–628 (2010)

PSOslope: A Stand-Alone Windows Application for Graphical Analysis of Slope Stability

Walter Chen[1] and Powen Chen[2]

[1] Dept. of Civil Engineering, National Taipei University of Technology, Taipei, Taiwan
waltchen@ntut.edu.tw
[2] Dept. of Civil Engineering/Institute of Information and Logistics Management, National Taipei University of Technology, Taipei, Taiwan
powellchen@gmail.com

Abstract. Landslide is an increasing problem and a cause for concern in densely populated areas. In addition to field studies and laboratory experiments, engineers also embrace computer technology to find better solutions and to achieve better landslide analysis. Traditionally, the main analysis framework is the limit equilibrium analysis, in which the state of limit equilibrium in the target slope is assumed to be reached along the entire sliding surface at the same time of failure. However, the assumption of circular sliding surfaces would prevent the search of non-circular sliding surfaces with lower factors of safety. To address this problem, particle swarm optimization (PSO) is implemented in this study using a computer program written in C# to automatically discover the optimal results in the target function. The results show that the PSO-based approach offers many interesting outcomes.

Keywords: Slope stability, sliding surfaces, landslides.

1 Introduction

Landslide is a naturally occurring disaster that occurs in all part of the world. In remote locations, landslide is a part of natural geological process causing little or no damage to human beings. However, when landslide takes place in populated areas, it can affect neighborhoods, communities, and cities causing large casualties and damage to infrastructures. With the world's rapid population growth and the current optimistic economic development trend, people are migrating to cities in search of a better life. Landslide is therefore an increasing problem and a cause for concern in densely populated areas near mountains such as the Taipei basin in Taiwan, where the Taipei city is located.

EM-DAT, the international disaster database was created by the WHO Collaborating Centre for Research on the Epidemiology of Disasters (CRED) and the Belgian Government in 1988 tracking major disasters in the world from 1900 to present [1]. There are 12 disaster types and more than 30 sub-types in the database, and landslides are classified under the "mass movement dry" or the "mass movement wet" categories. As an example of the seriousness of the landslide problem, Table 1 shows the

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 56–63, 2011.
© Springer-Verlag Berlin Heidelberg 2011

top 10 most important mass movement wet disasters for the period 1900 to 2011 sorted by the numbers of persons killed at the country level. It is worth noting that two of the top 10 most deadly landslides happened in the last five years and both of them took place in Asia.

Table 1. Top 10 most deadly landslide disasters (mass movement wet) from 1900 to 2011. Landslide disasters classified under the category of mass movement dry are not included in this table.

Country	Date	No Killed
Soviet Union, Landslide	1949	12000
Peru, Landslide	12/1941	5000
Honduras, Landslide	20/09/1973	2800
Italy, Landslide	9/10/1963	1917
China P Rep, Landslide	7/08/2010	1765
Philippines, Landslide	17/02/2006	1126
India, Landslide	1/10/1968	1000
Colombia, Landslide	27/09/1987	640
Peru, Landslide	18/03/1971	600
China P Rep, Landslide	23/03/1934	500

Created on: Jan-2-2011. - Data version: v12.07
Source: "EM-DAT: The OFDA/CRED International Disaster Database
www.em-dat.net - Université Catholique de Louvain - Brussels - Belgium"

2 Slope Stability Analysis

In order to combat the threat of landslides in landside-prone areas, efforts are being made to understand the mechanism of earth movement and more recently the role plants play in preventing shallow landslides and surface erosions [2]. In addition to field studies and laboratory experiments, engineers also embrace computer technology to find better solutions and to achieve better landslide analysis. Traditionally, the main analysis framework is the limit equilibrium analysis, in which the state of limit equilibrium in the target slope is assumed to be reached along the entire sliding surface at the same time of failure. More often than not, the sliding surface is further assumed to be circular for homogeneous and isotropic soils. Given the sliding boundary and the soil properties, both the driving forces (from gravity) and the resistance forces (from soil cohesion and friction) can be calculated, and the factor of safety (FS) of the given slope is defined as:

$$FS = \frac{Resistance\ Force}{Driving\ Force} \tag{1}$$

The FS is the de facto criteria used in assessing the stability of a slope. To remain stable, the FS of a slope has to be greater than one and is normally in the range of 1.1-1.5 to account for the uncertainty in the analysis and the rise of groundwater table or the possible additional earthquake force. To complete the analysis, the only thing left is to randomly or systematically select many circular arcs that pass through the slope and to locate the one with the lowest factor of safety (FS). The resulting sliding

surface of the lowest FS is the critical sliding surface and is considered to be the location where the failure takes place. However, the assumption of circular sliding surfaces would prevent the search of non-circular sliding surfaces with even lower FS. To address this problem, particle swarm optimization (PSO), an artificial intelligence technique for automatically discovering the optimal result in the target function, is implemented in this study using a computer program written in C# as discussed in the following sections.

3 PSO Formulation and Application Development

Particle Swarm Optimization (PSO) is an artificial intelligence method used to simulate the social behaviors of a group of animals such as birds and fishes and was proposed by Eberhart and Kennedy [3-4]. The basic idea is to consider a group of birds (agents or particles as referred to in this study) searching for the food (target or the minimum value of the optimization function as referred to in this study) in a 2-D space. Each bird (particle) remembers the closest point to the food (minimum value of the optimization function) that it has experienced before (called pbest) as well as the closest point to the food that any bird has experienced before (called gbest). Because the agents in the group exchange information continuously in order to achieve the common goal, the wisdom of crowds soon leads the group to converge on the target using the simple equation below:

$$v = v + 2 * r1 * (pbest - x) + 2 * r2 * (gbest - x) \quad (2)$$

where v and x are the velocity and the position of the individual agent respectively, and r1 and r2 are random numbers. In this study, PSO is used to solve the optimization problem in 2-D slopes. The goal is to determine the location of the sliding surface (arc or line segments) that has the lowest value of FS. To implement a solution using computer programs, a Windows stand-alone application is developed using Microsoft Visual C# in this study as shown in Figure 1, and the application is named PSOslope. When PSOslope is executed, a graphical user interface (GUI) will appear which can be roughly divided into two parts: the upper part shows the X and Y coordinates and a graphical representation of the slope profile, whereas the lower part provides input boxes to control the various parameters of the application. The input boxes can be further divided into five groups based on their intended functions. The first group from the left controls the definition of the slope profile. Four points (with X and Y coordinates) are allowed to define a given slope from left to right. Each time a different number is entered to the input boxes, the slope profile in the upper part of the window will be refreshed and redrawn immediately.

The second group of input boxes provides the initial values of the PSO search algorithm. For slope stability analysis, it simply defines the initial sliding surface using a circular arc, and all of the particles are located along this arc. The circular arc is a part of a circle whose center lies above the slope. The numbers to be filled into the input boxes are the X and Y coordinates of the center and the radius of the circle. If the user finds the circular sliding surface to be too restrictive, there is also a check box in this group to enable the random selection of the initial sliding surface when it is checked.

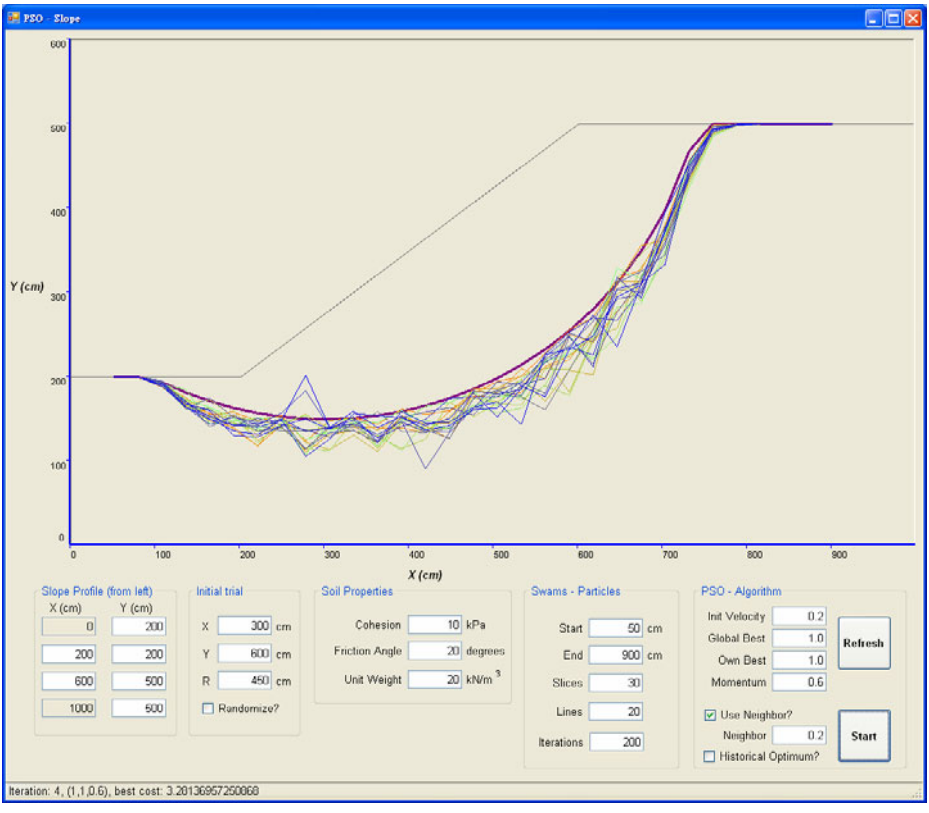


Fig. 1. PSOslope is a stand-alone program for slope stability analysis using the PSO technique. Launching the program will display a GUI window with five groups of input boxes on the lower part of the window to control the parameters of the application: slope profile, initial trial, soil properties, swarms, and PSO algorithm. The drawing on the upper part of the window represents the slope and will be refreshed and redrawn continuously during the program's execution.

The third group of input boxes presented to the user is the soil properties group. The soil that forms the slope is considered to be homogeneous and its shear strength parameters (cohesion and friction angle) and unit weight are entered here. Note that the metric units are used in this program. Using the ordinary method of slices, the final equation of the FS of the slope can be derived [5] from equation (1) and becomes equation (3):

$$FS = \frac{\sum [c_i l_i + W_i \cos \alpha_i \tan \phi]}{\sum W_i \sin \alpha_i} \tag{3}$$

where c = cohesion, ϕ = friction angle, l = the length of the slice base, α = the angle of the slice base, and W = the soil weight of the slice.

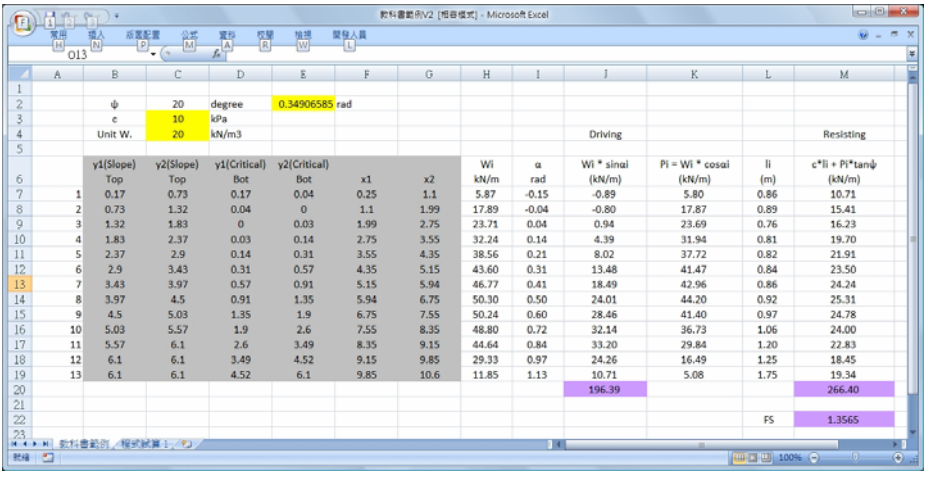
The fourth group of input boxes is used to specify parameters related to swarms or particles. The start and end boxes denote the left-most and the right-most positions,

respectively. Within this range, the slope is divided into some number of equal-width slices. The number is controlled by the input box labeled “slices.” The user can also specify the number of lines (i.e., the number of agents) and the number of iterations used in searching of the optimal solution (i.e., the minimum value of FS).

Finally, the last group of input boxes is used to specify the initial velocity, the momentum, and the coefficients of pbest and gbest used in the PSO equations. Pressing the start button will initiate the computation, and the screen will be refreshed and redrawn continuously. The user will notice many colored lines (representing the sliding surfaces) moving up and down in the slope until the optimal solution is found or the maximum number of iterations is reached. This interactive display of the searching process gives the user a very good visual cue and impression of the underlying activities of computation, which also sets this program apart from other programs.

4 Verification Assessment

The PSOslope program is checked to ensure the computational implementation is correct, and it can be applied to the analysis of real slopes. A spreadsheet program (Figure 2) is written to verify the computational results of PSOslope. The correctness of the spreadsheet program is in turn verified by a textbook example. The necessary two-step process is due to the fact that the PSOslope program only seeks out the critical sliding surface. It does not compute the FS of a pre-defined sliding plane. Therefore, the correctness of the PSOslope program to perform slope stability analysis and to compute the FS has to be checked indirectly. In this study, the authors wrote a spreadsheet program using equation (3) to compute the FS of the example 17-5 in reference [5], and found the results to be identical. Then, the PSOslope program was used to analyze a different slope. After the minimum FS was found by PSOslope, the coordinates of the critical sliding surface were outputted to a text file, and the text file was imported into the previous spreadsheet program to compute the corresponding FS. Both the PSOslope and the spreadsheet produced the same results.



		ψ	20	degree	0.34906585	rad							
		c	10	kPa									
		Unit W.	20	kN/m3						Driving			Resisting
		y1(Slope)	y2(Slope)	y1(Critical)	y2(Critical)			Wi	α	Wi * sinαi	Pi = Wi * cosαi	li	c*li + Pi*tanψ
		Top	Top	Bot	Bot	x1	x2	kN/m	rad	(kN/m)	(kN/m)	(m)	(kN/m)
	1	0.17	0.73	0.17	0.04	0.25	1.1	5.87	-0.15	-0.89	5.80	0.86	10.71
	2	0.73	1.32	0.04	0	1.1	1.99	17.89	-0.04	-0.80	17.87	0.89	15.41
	3	1.32	1.83	0	0.03	1.99	2.75	23.71	0.04	0.94	23.69	0.76	16.23
	4	1.83	2.37	0.03	0.14	2.75	3.55	32.24	0.14	4.39	31.94	0.81	19.70
	5	2.37	2.9	0.14	0.31	3.55	4.35	38.56	0.21	8.02	37.72	0.82	21.91
	6	2.9	3.43	0.31	0.57	4.35	5.15	43.60	0.31	13.48	41.47	0.84	23.50
	7	3.43	3.97	0.57	0.91	5.15	5.94	46.77	0.41	18.49	42.96	0.86	24.24
	8	3.97	4.5	0.91	1.35	5.94	6.75	50.30	0.50	24.01	44.20	0.92	25.31
	9	4.5	5.03	1.35	1.9	6.75	7.55	50.24	0.60	28.46	41.40	0.97	24.78
	10	5.03	5.57	1.9	2.6	7.55	8.35	48.80	0.72	32.14	36.73	1.06	24.00
	11	5.57	6.1	2.6	3.49	8.35	9.15	44.64	0.84	33.20	29.84	1.20	22.83
	12	6.1	6.1	3.49	4.52	9.15	9.85	29.33	0.97	24.26	16.49	1.25	18.45
	13	6.1	6.1	4.52	6.1	9.85	10.6	11.85	1.13	10.71	5.08	1.75	19.34
										196.39			266.40
												FS	1.3565

Fig. 2. A spreadsheet program is written to verify the computational results of PSOslope. The correctness of the spreadsheet program is in turn verified by a textbook example.

5 Computational Findings and Discussion

After the computational implementation of PSOslope was verified, it was then possible to explore the answers obtained by the program. For comparison purposes, the target slope (Figure 1) was first analyzed by the STABL program developed by Purdue University to obtain a reference FS of 1.957. Recall from previous sections that the PSO algorithm no longer restricts the sliding surfaces to be circular. As a result, PSOslope was able to converge to solutions with unexpected shapes as shown in Figures 3, 4, and 5 (FS = 1.680 to 1.871) in addition to the usual circular shape as shown in Figure 1. In fact, because of the complexity of the object function (equation 3), there seems to be many local minimum solutions that can be discovered by the PSO algorithm, and Figures 3, 4, and 5 are not isolated cases. Rather, each of them represents a group of solutions with similar shapes. What's more, sometimes the entire minimal FS searching process could fall apart under unfavorable conditions (unlucky random numbers) as illustrated in Figure 6, and the computation simply failed to converge to a stationary value and produced a large negative FS. As the slope under consideration is sloping to the left, the solutions with negative FS values are easy to be ruled out because they would represent the calculated stability values if the slope were sliding (or pushed) to the right. In contrast, the solutions characterized by Figures 3, 4, and 5 seem possible at first glance. Does it imply that every slope like this one may contain many possible local optimal solutions that haven't been carefully examined? Are these solutions not "kinematically admissible" as suggested by some researchers [6]? What if the sliding surfaces in Figures 3, 4, and 5 are further smoothed to remove the sharp corners (points) where derivatives do not exist? To answer these questions and to determine whether a slope movement is kinematically admissible seems to require more investigations of the different types of sliding surfaces. Since there is yet to be a rigorous proof of the "kinematically admissible" assertion, it is probably better to avoid drawing any conclusions based on intuition alone and defer judgment until more study results become available.

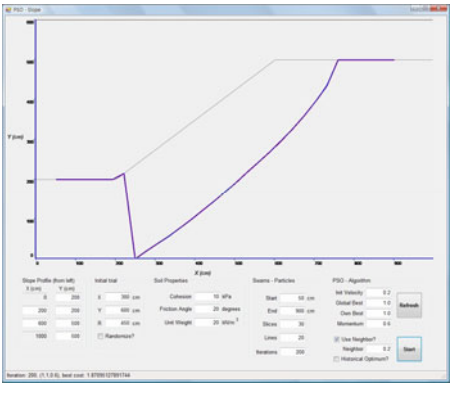


Fig. 3. Possible type of sliding surface #1

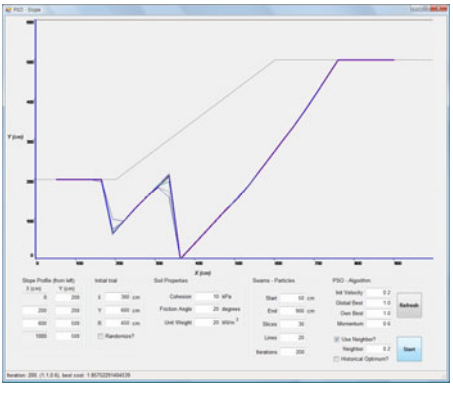


Fig. 4. Possible type of sliding surface #2

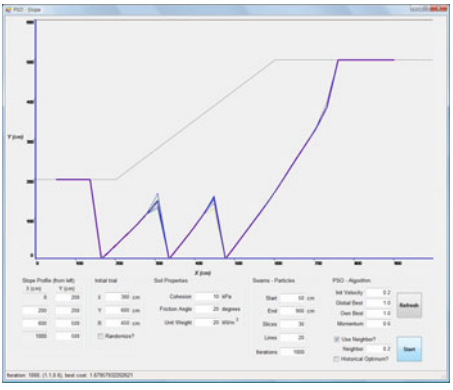


Fig. 5. Possible type of sliding surface #3

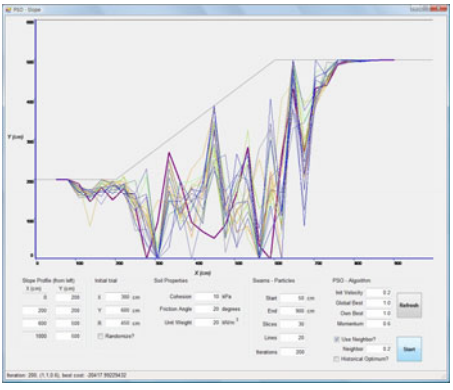


Fig. 6. Computation fails to converge to a stationary value

6 Summary and Conclusions

Around the world and in Taiwan, there are so many landslide disasters occurring every year that the thorough understanding of the landslide mechanism is considered as one of the most actively researched upon subjects in geotechnical engineering. With so many researches and case studies, the ability to integrate the research findings and explore novel options may be the most critical determinant of what can be achieved working together in this particular research field. In this paper, the authors investigated the possibility of using PSO to better determine the location of sliding surfaces. A computational solution based on the PSO algorithm under the analysis framework of limit equilibrium was implemented using C#, and a stand-alone Windows application (PSOslope) was developed for the graphical analysis of slope stability (calculation of FS). In addition, a spreadsheet program was written and a textbook example was used to verify this study's computational results. The authors further compared the answers obtained by PSOslope with those obtained by Purdue University's STABL program. The results indicated that the PSO-based approach offered many surprising outcomes and interesting issues to be further studied. For example, of particular significance in the light of this study are the choice of parameters and the sensitivity of results in the PSO model. It is believed that the interactive display of PSOslope will provide users immediate feedback on their analysis (not available in the past), and therefore will become a valuable tool in the future analysis of slope stability problems.

Acknowledgments. This study was partially supported by grant numbers NSC 99-2218-E-027-008, NSC 98-2221-E-027-095, and NSC 97-2221-E-027-091 from the National Science Council of Taiwan (ROC).

References

1. EM-DAT, The International Disaster Database, http://www.emdat.be/
2. Chen, W.W., Lin, J.-T., Lin, J.-H., Shen, Z.-P.: Development of the Vegetated Slope Stability Analysis System. J. of Software Engineering Studies 4(1), 16–25 (2009)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE International Conf. on Neural Networks, pp. 1942–1948. IEEE Service Center, Los Alamitos (1995)
4. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. Sixth Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Service Center, Los Alamitos (1995)
5. Salgado, R.: The Engineering of Foundation. McGraw Hill, Boston (2008)
6. Cheng, Y.M., Li, L., Chi, S., Wei, W.B.: Particle Swarm Optimization Algorithm for the Location of the Critical Non-circular Failure Surface in Two-dimensional Slope Stability Analysis. Computers and Geotechnics 34, 92–103 (2007)

A Review of the Application of Swarm Intelligence Algorithms to 2D Cutting and Packing Problem

Yanxin Xu, Gen Ke Yang, Jie Bai, and Changchun Pan

Department of Automation, Shanghai JiaoTong University and Key Laboratory of System Control and Information Processing, Ministry of Education of China, 800 DongChuan Rd, MinHang District, Shanghai, China,
{iamxuyanxin,gkyang,baijie,pan_cc}@sjtu.edu.cn

Abstract. Cutting and packing (C & P) problem is to allocate a set of items to larger rectangular standardized units by minimizing the waste. Bin packing, strip packing and cutting stock problem is well-known classical C & P problem. An overview is provided of several meta-heuristics algorithms of swarm intelligence from the literature for the 2D C & P problem. The objective of this paper is to present and categorize the solution approaches in the literature for 2D regular and irregular C & P problem. The focus is hereby on the analysis of the methods and application of swarm intelligence algorithms.

Keywords: cutting and packing problem, swarm intelligence, ACO, PSO.

1 Introduction

Cutting and packing (C & P) problems are optimization problems concerned with finding a good arrangement of multiple items in larger containing regions. There are many classic cutting and packing problems, including the cutting stock problem, the bin packing problem, the strip packing problem etc. This type of problem arises in a wide variety of industries, such as garment manufacturing, sheet metal cutting, furniture making and shoe manufacturing. Figure 1 provides an example of a layout from the garment manufacturing industry. High material utilization is of particular interest to mass production industries since small improvements of the layout can result in large saving of material and considerably reduce production cost. The objective of the packing process is to maximize the utilization of material. The manual generation of layouts is costly in terms of man-power hence methods for the automation of packing are being sought.

In this paper, we focus our attention on meta-heuristic methods to solve the 2D C & P problem. Particular emphasis is put on solutions involving swarm intelligence, which is increasingly be utilized in combinational optimization problems. In the last two decades, the computational researchers have been increasingly interested to the natural sciences, and especially biology, as source of modeling paradigms. Many research areas are massively influenced by the behavior of various biological entities and phenomena. It gave birth to most of population-based Meta-heuristics such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) etc. They modeled

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 64–70, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the insect social behaviors such as ant, fish, bird, bee etc. They could be regarded as belonging to the category of intelligent optimization tools used to solve a computational and complex problem in different areas [1].

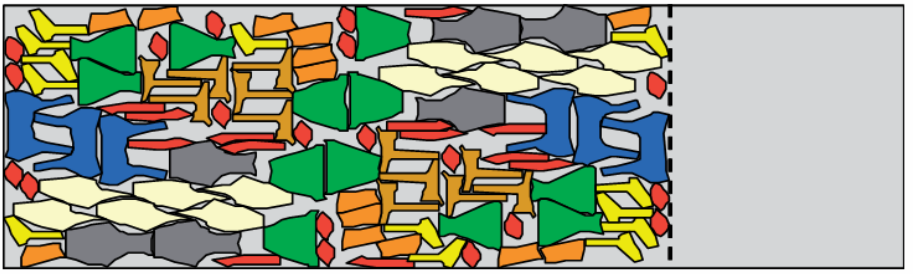


Fig. 1. An example layout from garment manufacturing

The 2D C & P problem has been shown to be NP-hard and is therefore intrinsically difficult to solve (Garey and Johnson 1979 [2]). Therefore, many solution methods have been proposed for the problem, but so far an algorithm that would yield the optimum solution might not exist; thus, general methods are intended to find an acceptable approximate solution in short time. Many approaches have been proposed in the literature. A recent survey of the regular shapes packing problems were given by Lodi, Martello, and Monaci (2002) [3], the more complete revision has been presented in Hopper 2000 [4]. However in the last ten years the interest in this subject has increased, especially in swarm intelligence aspect, as well as the number of papers presenting new approaches and improvements to existing strategies. We review here the most recent results in this research area.

This paper is organized as follows: in the next section, we give an introduction of the 2D C & P problem. In section three, we review the methods based on metaheuristics, especially the swarm intelligence on 2D C & P problem. Section four is related to discussion about analysis and comparatives on the introduced methods. Finally, section five presents the conclusion and future trends in this research area.

2 Two-Dimensional Cutting and Packing Problem

C & P problem include cutting and packing problem respectively. In the case of packing problems the large objects are defined as empty and need to be filled with small items. Cutting problems are characterized by large objects that need to be cut up into small items. Dyckhoff (1990) [5] emphasizes the strong relationship between cutting and packing problems. In this sense cutting stock problems can be translated into packing problem. Therefore, we focus on packing problem in the following sections.

C & P problem can be classified according to different characteristics of objects, such as the shape, the number, and their orientation. Dyckhoff proposed a useful classification of C & P problems, which was revised recently by W¨ascher et al [6]. Their classification partitioned the problems by dimensionality, objective of the assignment,

large objects and small items. The new typology provides a useful progression towards a consistent nomenclature for cutting and packing problems [7].

Concerning the geometry of objects and items, two types of problems can be distinguished: regular (rectangles, circles) and irregular (asymmetries and concavities) C & P problem. Furthermore, two main types of packing problem, strip packing and bin packing, can be distinguished depending on the type of large object. In the paper and textile industry the raw material is available in the form of rolls. Hence the packing process aims at reducing the height of the layout. Bin packing refers to packing of multiple bins and can be found where the stock material is available in the form of sheets. The objective usually is to find the set of bins to accommodate all parts of the order list under minimization of the total material used.

The model of the 2D rectangular strip packing problem can be formulated as follows:

$$\begin{aligned} &Minimize \ \ H \\ &Subject\ to \\ &x_i + w_i \le W, \forall i \in I, \end{aligned} \tag{1}$$

$$y_i + h_i \le H, \forall i \in I, \tag{2}$$

$$x_i + w_i \le x_j, or, x_j + w_j \le x_i, or, \tag{3}$$

$$y_i + h_i \le y_j, or, y_j + h_j \le y_i, \forall i, j \in I, i \ne j, \tag{4}$$

$$x_i, y_i \ge 0, \forall i \in I. \tag{5}$$

Where, H is the Height of the strip, which is minimized, W is a fixed width of the strip. Each object i has dimensions h_i (height) and w_i (width), i=1,……n. The Cartesian coordinates of (x_i,y_i) is the location of the bottom left corner. A feasible solution must fulfill the following constraints: the object must be within the rectangular area, and must not overlap with any other object [8].

The solution method of C & P problem can be classified into complete and approximation approaches. The complete approaches include mathematical programming, such as linear programming, dynamic programming, column generation and branch and bound etc. The approximation approaches are mainly heuristics, which are some heuristic packing rules, and meta-heuristics such as ACO and PSO etc. Since search space of C & P problem is increasingly larger and larger, in order to search for their global optimal solutions, complete approaches take too much computational time to solve the problem. Various meta-heuristic algorithms, especially swarm intelligence algorithms, have been adopted as optimization tools to find good solutions fast. In recent years, it has become evident that the skilled combination of a meta-heuristic with other optimization techniques, a so called hybrid meta-heuristic, can provide a more efficient behavior and a higher flexibility when dealing with real-world and large-scale problems [9]. Some of the hybrid meta-heuristics approach will be introduced in the following section.

3 Applications of Swarm Intelligence Algorithms in 2D Cutting and Packing Problem

The 2D C & P problem is NP hard due to the combinatorial explosion encountered as the problem size increases. As a result, published solution approaches focus on heuristic and meta-heuristics methodologies. Meta-heuristics are general frameworks for heuristics in solving combinatorial optimization problems. Although there are many different solution approaches presented in the literature, there appear to be two key strategies for representing and searching the solution space. The first approach is to represent the solution as an ordered list of pieces and apply a placement rule to construct the solution. The second approach represents the solution as a physical layout on the stock sheet and moves pieces within the layout [10]. This paper focuses on the former strategy.

The former strategy is dependent on two critical characteristics of the algorithm; the placement rule and the placement order of the pieces. It's proved that better solutions are obtained by performing local search over the permutation and repeatedly constructing solutions. The majority of these literature meta-heuristics concentrates on hybrid algorithms, where a meta-heuristics is combined with a heuristic placement routine. The sequence of the item to be packed is usually constructed as a sequence chromosome, which is decoded via a packing heuristic for generating a packing plan to be evaluated against the objective function to obtain the quality of the solution. The main meta-heuristics utilized in the first approach are EA, TS, SA, ACO, and PSO etc. The applications of these meta-heuristics are introduced respectively as followed.

3.1 Applications of Ant Colony Algorithms

ACO is one of the most recent techniques for approximate optimization. The development of these algorithms was inspired by the observation of ant colonies. The inspiration for ACO is the ants' foraging behavior, and in particular, how ants can find shortest paths between food sources and their nest. The key to this ability lies in the fact that ants leave a pheromone trail behind while walking. The pheromone trails will guide other ants to the food source. This characteristic of real ant colonies is exploited in ACO algorithms in order to solve, for example, discrete optimization problems [11]. It is widely applied in integer linear programming, continuous optimization problem and clustering problem etc.

Ducatelle and Levine 2002 [12] adopted ACO to solve bin packing and employed artificial ants to build solutions stochastically, with heuristic information obtained by First Fit Decreasing (FFD) and an artificial pheromone trail, which is encoded by the favorability of having two items in the same bin. This approach may get steady performance and a good solution even not finding the optimum. However, its computing time is longer than others. As a result, Levine and Ducatelle 2004 [13] improve the performance by combing ACO and an iterated local search approach, since a local search algorithm can greatly improve the performance of an ACO approach. The hybrid approach they proposed to the bin packing is quite generic, and could be capable of being adapted to solve similar grouping problems.

Thiruvady[14] employed a hybrid approach by combining ACO and Bottom-Left-Fill (BLF) to solve rectangular strip packing problem. The ordering and orientations of items are generated by ACO and the items are allocated one by one with BLF heuristic according to the ordering and orientations given by ACO. They compared and analyzed four combinations between ordering and orientation. The results showed that packing items with learning ordering and orientations obtained by ACO outperformed the other combinations.

Yi-Chun Xu. et.al [15] proposes a constructive heuristic to pack weighted items in a circular container. They use an ant-based algorithm and optimize the packing order with the base of this heuristic. In their ant algorithm, encode of pheromone matrix considers the favorability of choosing an item and the product of the size and the weight of the next packed item. By doing so, large and heavy items have higher priority and this is beneficial for the packing performance. They also compared two versions of the ant-based algorithm, AS and Min-Max AS, with existing approaches, such as the genetic algorithm, and the hybrid PSO. However, since their research only focus on the regular objects (circle and rectangle), it's a problem for them whether their approach can obtain good performance for irregular shaped objects.

3.2 Applications of Particle Swarm Optimization

PSO is firstly introduced by Kennedy and Eberhart in 1995, which was originated from the simulation of behavior of bird flocks. In PSO a number of simple entities—the particles—are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function [16]. It is widely applied in continuous optimization, combinational optimization problem and clustering problem etc.

The applications of PSO are considerably less in 2D C & P problem. D. S. Liu et al 2006 [17] presented a two-objective mathematical model with multiple constraints for two dimensional bin packing problem and solve the problem with a hybrid multi-objective PSO algorithm. They choose BLF heuristics as the decoding heuristic since it has the ability to fill in the gaps in the partial layout. They created variable length data structure and specialized mutation operator to make hybrid multi-objective PSO algorithm as a robust search optimization algorithm. Liu et al 2008 [18] made some improvements on the basis of Liu et al 2006 and proposed a multi-objective model and a PSO-based algorithm which incorporates EA concepts such as the use of mutation operator as a source of diversity. With the use of hybrid approach combining evolutionary PSO and BLF heuristic, they solve bin packing problem effectively and efficiently. The performance shows their method outperforms the one in Liu et al 2006 and is also applicable to single-objective bin packing problem.

4 Discussion

Since a couple of researches have adopted meta-heuristic algorithms such as EA, GA, SA, TS and ANN to solve C & P problems, swarm intelligence algorithms are relatively less utilized to this field. However, swarm intelligence algorithms are beginning to attract more and more interest of researchers for their characteristics. The goal of swarm intelligence is the design of intelligent multi-agent systems by taking inspiration from the collective behavior of social insects such as ants, termites, bees, wasps, and other animal societies such as flocks of birds or fish schools.

ACO has been applied in almost all kinds of C & P problems including strip packing, bin packing and irregular packing problems. The principle of ACO is positive feedback mechanism and ant colony can be regarded as a reinforcement learning system. Its character of positive feedback and synergy bring ACO good performance in distributed system.

PSO is only applied in bin packing problem so far. Although PSO is still lack of rigorous mathematical foundation in mathematical analysis, especially in convergence analysis, the characteristics of fast convergence and parallel computing make PSO widely applied in engineering applications.

Although ACO and PSO get relatively less attention for their date births, they are still widely applied in many respects of optimization problems for their advantages introduced above. However ACO and PSO also have some shortcomings in some respects. Almost all the articles introduce above make a comparison between ACO or PSO and EA, EP or GA, the other meta-heuristics. From the results of comparisons, we can find that swarm intelligence and other meta-heuristics have their own advantages respectively. Swarm intelligence algorithms outperform other meta-heuristics such as EA, GA, EP etc in some cases and vice versa. They may also get similar performance in some cases. It's easy to understand that the use of any algorithm has its conditions and background. Therefore, it's a trend to combine different algorithm to give full play to their advantage and make up the deficiency. Thus the hybrid algorithm, such as hybrid algorithm with ACO and GA, is able to get better performance than any single algorithm, which get proof from the articles introduced above.

5 Conclusion

This paper presents an overview of the recent advances of swarm intelligence algorithms in solving the two-dimensional cutting and packing problem. Some articles about C & P problems which adopted ACO and PSO are introduced in this paper. The preliminary discussion about swarm intelligence and other meta-heuristics algorithms is also proposed. It is not the aim of this review to try and cover all the various elements discussed in the literature relevant to cutting and packing problem. The aim of this review is to concentrate on the elements in the literature that are relevant to swarm intelligence algorithms.

We are interested in Meta-heuristics, including swarm intelligence methods. In order to increase the e1ectiveness of the meta-heuristics approaches, especially those based on swarm intelligence we can then conclude for our further researches will both

focus on designing and utilizing more swarm intelligence meta-heuristics and improve the performance of existed ones to solve cutting and packing problems.

Acknowledgments. The work of this paper was supported by National Natural Science Foundation of China, No.61074150, and No.60574063.

References

1. Bitam, S., Batouche, M., Talbi, E.g.: A survey on bee colony algorithms. In: The 24th IEEE Parallel & Distributed Processing Symposium Workshops (IPDPSW), Georgia, pp. 19–23 (2010)
2. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
3. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research 141, 241–252 (2002)
4. Hopper, E., Turton, B.C.H.: A review of the application of meta-heuristic algorithms to 2D strip packing problems. Artificial Intelligence Review 16, 257–300 (2000)
5. Dyckhoff, H.: A typology of cutting and packing problems. European Journal of Operational Research 44, 45–159 (1990)
6. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J. Oper. Res. 183, 1109–1130 (2007)
7. Bennell, J.A., Oliveira, J.F.: A tutorial in irregular shape packing problems. Journal of the Operational Research Society 60, 93–105 (2009)
8. Riff, M., Bonnaire, X., Neveu, B.: A revision of recent approaches for two-dimensional strip-packing problems. Engineering Applications of Artificial Intelligence 22, 823–827 (2009)
9. Hopper, E., Turton, B.C.H.: An empirical investigation on meta-heuristic and heuristic algorithms for a 2d packing problem. European Journal of Operational Research 128, 34–57 (2001)
10. Bennell, J.A., Song, X.: A beam search implementation for the irregular shape packing problem. J. Heuristics 16, 167–188 (2008)
11. Blum, C.: Ant colony optimization: Introduction and recent trends. Phys. Life Reviews 2, 353 (2005)
12. Ducatelle, F., Levine, J.: Ant Colony Optimization for Bin Packing and Cutting Stock Problems. In: UK Workshop on Computational Intelligence, Edinburgh (2001)
13. Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. J. Oper. Res. Soc. 55, 705 (2004)
14. Thiruvady, D.R., Meyer, B., Ernst, A.T.: Strip packing with hybrid ACO Placement order is learnable. In: IEEE Congress on Evolutionary Computation, Hong Kong, pp. 1207–1213 (2008)
15. Xu, Y.C., Dong, F.M., Liu, Y., Xiao, R.B.: Ant Colony Algorithm for the Weighted Item Layout Optimization Problem. eprint arXiv:1001.4099 (2010)
16. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization: an overview. Swarm Intelligence Journal 1(1), 33–57 (2007)
17. Liu, D.S., Tan, K.C., Goh, C.K., Ho, W.K.: On solving multi-objective bin packing problems using particle swarm optimization. In: IEEE Congress on Evolutionary Computation, Vancouver, pp. 7448–7455 (2006)
18. Liu, D., Tan, K., Huang, S., Goh, C., Ho, W.: On solving multi-objective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research 190, 357–382 (2008)

Inertia Weight Adaption in Particle Swarm Optimization Algorithm*

Zheng Zhou[1] and Yuhui Shi[2]

[1] Dept. of Computer Scienceand Software Engineering,
Xi'an Jiaotong-Liverpool University, Suzhou 215123, China
zheng.zhou09@student.xjtlu.edu.cn
[2] Dept. of Electrical and Electronic Engineering,
Xi'an Jiaotong-Liverpool University, Suzhou 215123, China

Abstract. In Particle Swarm Optimization (PSO), setting the inertia weight *w* is one of the most important topics. The inertia weight was introduced into PSO to balance between its global and local search abilities. In this paper, first, we propose a method to adaptively adjust the inertia weight based on particle's velocity information. Second, we utilize both position and velocity information to adaptively adjust the inertia weight. The proposed methods are then tested on benchmark functions. The simulation results illustrate the effectiveness and efficiency of the proposed algorithm by comparing it with other existing PSOs.

Keywords: PSO, inertia weight, velocity information, adaption.

1 Introduction

Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart in 1995[1][2]. PSO uses a simple mechanism that simulates swarm behavior in birds flocking to search for globally optimal solution. PSO has been rapidly developed in recent years due to its simplicity in concept and easiness in implementation.

In PSO, a population of particles represents a group of candidate solutions to the problem to be solved. Each particle is associated with two vectors, the position tor$X_i = [\, x_i^1, x_i^2, \ldots, x_i^D]$ and the velocity vector$V_i = [v_i^1, v_i^2, \ldots, v_i^D]$, where D means that the solution space is a D-dimensional space. The basic equations are usually given as follows: [1] [2]:

$$v_{id} = w * v_{id} + c_1 * r_{1d} * (pBest_{id} - x_{id}) + c_2 * r_{2d} * (gBest_{id} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

In equation (1), w is known as inertia weight [3]; c_1, c_2 are acceleration coefficients; r_{1d} and r_{2d} are two random values in the range[0, 1].

* This work is supported by National Natural Science Foundation of China under Grant No.60975080, and Suzhou Science and Technology Project under Grant No. SYJG0919.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 71–79, 2011.
© Springer-Verlag Berlin Heidelberg 2011

The inertia weight w was first introduced to linearly decrease [3] over generations as shown in equation (3).

$$w = w_{\max} - (w_{\max} - w_{\min}) \frac{gen}{GENERATION} \tag{3}$$

where *gen* is the current generation number and *GENERATION* is the maximum number of generations. The range of inertia weight w is [0.4, 0.9] which can adjust the local and global search ability [3] [4].

Another improvement is the introduction of constriction factor to PSO by M. Clerc [5]. The constriction factor replaces the equation (1) with equation (4) and (5).

$$v_{id} = K[w * v_{id} + c_1 * r_{1d} * (pBest_{id} - x_{id}) + c_2 * r_{2d} * (gBest_{id} - x_{id})] \tag{4}$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, where \varphi = c_1 + c_2, \varphi > 4 \tag{5}$$

Where K is set to 0.729 and c_1, c_2 are both 2.05.

Adaptive PSO (APSO) was proposed by Z. Zhan and J. Zhang in 2009 [6]. In APSO, the inertia weight, acceleration coefficients is controlled automatically. Furthermore, an elitist learning strategy (ELS) is applied in convergence state to avoid being stuck into local optimal. The APSO utilized only the particles' position information. As we knew, PSO is different from other evolutionary algorithms in that each individual (particle) is also associated with a velocity in addition to the position. Therefore, it is natural to believe that an adaptive PSO, which utilize both position and velocity information, maybe can improve PSO' performance further.

In this paper, first, we will introduce a method which borrows the idea in APSO except that the inertia weight will be adaptively adjusted based on velocity information instead of position information in APSO. The purpose of doing so is to verify whether velocity information alone can also be utilized to improve PSO's performance. Then we will utilize both position and velocity information to adaptively adjust PSO's inertia weight. The remaining paper is organized as follows, methods to adaptively control the inertia weight will be proposed in Section 2. In Section3, the PSO proposed will be tested and compared with other representative PSOs. Finally, conclusions are summarized in Section 4.

2 Adaptive Control of Inertia Weight

2.1 Methods in APSO

As we borrowed the idea from APSO, we first present the process of APSO as follows [6]:

Step 1: Initialize the particles' position and velocity as in traditional PSOs.

Step 2: For each particle *i,* calculate the mean distance to all the other particles by the equation (6) as below:

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_{ik} - x_{jk})^2} \tag{6}$$

Where N is the number of particles and D is the number of dimension.

Step 3: Set the global best particle's mean distance to all the other particles as d_g. Compare all d_i's, find the maximum and minimum distances d_{max} and d_{min}. Then determine an "evolutionary factor" f as:

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \tag{7}$$

Step 4: Classify the value of f into four sets which represent the states of exploration, exploitation, convergence and jumping out.

Step 5: Adjust the inertia weight by the equation (8), and adjust acceleration Coefficients in different state by the rules in table1.

$$w(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4,0,9] \tag{8}$$

Step 6: In the Convergence state, ELS approach is applied.

Table 1. Strategies for control c_1 and c_2

State	C_1	C_2
Exploration	Increase	Decrease
Exploitation	Increase slightly	Decrease slightly
Convergence	Increase slightly	Increase slightly
Jumping-out	Decrease	Increase

The APSO is proved to enhance the performance of PSO in convergence speed and global optimality [6]. In APSO, the inertia weight is adjusted based on the position information. In the following section, a velocity-based adaptive inertia weight will be introduced to verify the usefulness of velocity information.

2.2 Adaption of Inertia Weight Based on Velocity

The new method uses velocity information instead of the position information used in APSO. The mean velocity difference between each particle and all other particles can be calculated as in equation (9) which is similar to equation (6) in APSO.

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (v_{ik} - v_{jk})^2} \tag{9}$$

Where N is the population size and D is the number of dimension.

Then define a velocity-based evolutionary factor and inertia weight adaption by the equation (7) (8) used in APSO. For convenience, they are given as follows:

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \tag{10}$$

$$w(f) = \frac{1}{1+1.5e^{-2.6f}} \in [0.4,0,9]$$

In this paper, the w is kept in the range [0.4, 0.9] during the PSO process.

2.3 Adaptive Inertia Weight Based on Both Position and Velocity Information

As mentioned above, we expect that by using both position and velocity information to adjust the inertia weight can make the PSO perform better. A method implements this goal is described as follows:

Firstly, denote the position-based evolution factor in equation (8) and the velocity-based evolution factor in equation (10) as f_p and f_v, respectively.

Then define a new hybrid evolutionary factor by set two weights w_p and w_v to f_p and f_v as in the following equation (11):

$$f = w_p * f_p + w_v * f_v \tag{11}$$

Adjust the inertia weight by the same equation as (8) as below:

$$w(f) = \frac{1}{1+1.5e^{-2.6f}} \in [0.4,0,9]$$

The w_p and w_v can be set with different values. In our test, the w_p and w_v are both set to be 0.5, which means that the position and velocity information have the same influence on the inertia weight.

3 Experiments and Comparison

3.1 Benchmark Functions and Variants of PSOs

Six benchmark functions are used for the test, which are listed in Table2. These functions are widely used to test the evolutionary functions [7]. Among these functions, f_1, f_2, f_3 are unimodal functions and f_4, f_5, f_6 are multimodal functions.

For comparison, variants of PSOs are used in this paper. The GPSO is the one with linearly decreasing inertia weight from 0.9 to 0.4 proposed in [3]. The CFPSO is a PSO with constriction factor = 0.729 proposed in [5]. The APSO is proposed by Z. Zhan which was reviewed in Section 2 [6]. The last two PSOs are the adaptive inertia weight based on velocity information alone and based on both position & velocity information introduced in this paper, referred to as AIPSO (Adaptive velocity Inertia weight PSO) and AHPSO (Adaptive hybrid Inertia Weight PSO) respectively.

For fairness of the test, the PSOs are with the population size of 20, dimensions of 30 and the same number of 2×10^5 FEs is used for the test functions. In order to reduce statistical errors, each function will be tested for 30 times under each PSO algorithm, and the mean results will be used for the purpose of comparisons.

Table 2. Test Functions used in this paper

Test functions	Search Space	f_{min}	Acceptance	Function name
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100,100]^D$	0	0.01	Sphere
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10,10]^D$	0	0.01	Schwefel's P2.22
$f_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100,100]^D$	0	100	Quadric
$f_4(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2) + (x_i - 1)^2]$	$[-10.10]^D$	0	100	Rosenbrock
$f_5(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{x_i})$	$[-500,500]^D$	-12569.5	-10000	Schwefel
$f_6(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - -\prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	$[-600,600]^D$	0	0.01	Griewank

3.2 Comparison and Discussion

The solutions obtained by each PSO for different test functions are listed in Table 3.The boldface indicates the best results among the algorithms. The experiment results show that among GPSO, CFPSO, APSO and AIPSO, AIPSO can obtain the best solution on unimodal function f_1,f_2,f_3, but on multimodal functions *f4, f5, f6*, AIPSO does not perform the best.

Table 3. Search result comparison for variants PSOs

Function		GPSO	CFPSO	APSO	AIPSO	AHPSO
f_1	Mean	1.20e-53	4.47e-140	1.45e-150	**1.16e-159**	2.67e-159
f_2	Mean	7.93e-34	7.15e-30	5.13e-84	**7.07e-89**	2.19e-86
f_3	Mean	4.13e-2	3.85e-10	1.61e-10	**4.45e-11**	9.67e-11
f_4	Mean	34.61	**2.74**	3.49	11.64	9.14
f_5	Mean	-9719.71	-7038.52	**-12596.5**	-7326.27	-7348.93
f_6	Mean	1.56e-2	5.81e-2	1.66e-2	1.56e-2	**1.21e-2**

The results on uninodal functions indicate that the method proposed in this paper can improve the performance of PSO. In particular, the AIPSO perform best on unimodal functions f_1,f_2,f_3, and the standard deviation of the solutions is the lowest while AHPSO outperforms the APSO but lose to AIPSO by a small difference. For multimodal functions, the AHPSO can obtain the best solution on $f_6$but both AHPSO and AIPSO drop into local optimal on f_4 and f_5.

For the comparison of the algorithms' search speed, Table 4 lists the mean number of FEs (function evaluations) to obtain an acceptable solution. The CPU runtime is utilized to measure the computational process, because the PSOs used in this paper are different and some of them have extra calculations to increase the runtime.

It can be seen that AHPSO cost the least runtime onunimodal functionsf_1, $f_2$and AHPSO is the fastest. On multimodal functionsf_4,f_5,f_6, APSO cost the least CPU runtime and AHPSO ranked second on $f_4$andf_6. It should be noticed that the AIPSO and AHPSO did not reach the acceptable solution on $f5$and fall into local optimal.

Table 4. Number of FEs and mean run time needed to reach acceptable solution

Function		GPSO	CFPSO	APSO	AIPSO	AHPSO
f_1	Mean FEs	106191	199998	7099	7063	**6937**
	time(sec)	0.97	0.12	0.11	0.11	**0.09**
f_2	Mean FEs	103225	199981	7925	7921	**7343**
	time(sec)	1.02	0.22	0.17	0.13	**0.11**
f_3	Mean FEs	135797	199909	**21164**	22172	22061
	time(sec)	2.02	2.32	**0.98**	0.99	0.99
f_4	Mean FEs	101533	199995	**5334**	8150	5360
	time(sec)	0.91	0.99	**0.09**	0.11	0.09
f_5	Mean FEs	90548	---	**5159**	---	---
	time(sec)	1.81	---	**0.12**	---	---
f_6	Mean FEs	116235	11432	**7568**	8081	7938
	time(sec)	1.39	0.23	**0.16**	0.17	0.13

The comparison of FEs is much more interesting than CPU runtime. For instance, test on f_1 tell us that the mean number of FEs of 106191, 199998, 7099, 7063,6937 are needed by GPSO, CFPSO, APSO , AIPSO and AHPSO to reach an acceptable solution, respectively. Among these data, AHPSO only uses 6937 FEs and CPU runtime of 0.09 seconds, which is the best performance among all the PSOs.

In the above experiments, the results of GPSO, CFPSO, APSO, AIPSO and AHPSO are compared, we can see that the proposed adaptive inertia weight methods are efficient for PSO. The new PSOs outperform the other PSOs with a faster convergence speed and obtain better optimization results in most unimodal test functions but not the multimodal functions.

The comparison among APSO, AIPSO and AHPSO should be specially focused. In general, AISPO and AHPSO perform much better on unimodal functions and APSO perform better on multimodal functions. As APSO has adaptive inertia weight, adaptive acceleration coefficients and a complex ELS approach to help to jump out of local optima, while AIPSO and AHPSO only uses an adaptive inertia weight, we can say that the result of AISPO and AHSPO is acceptable and the proposed method which uses velocity information or both position and velocity information to replace position information to adjust inertia weight in PSOs is promising.

3.3 PSOs with Elitist Learning Strategy (ELS)

As the AISPO and AHPSO do not perform well on multimodal functions, we consider using Elitist Learning Strategy (ELS) as in APSO to helps PSO to jump out of local optimal in AIPSO and AHPSO. The details are given in the following.

The APSO [6] has an ingenious method named Elitist Learning Strategy (ELS) to help the particle swarm to jump out of local optimal. The ELS gives a random velocity generated from Gaussian distribution to the global best particle to push it to run for a potentially better region in the convergence state.

If the global best particle finds a better region, it will move there and all other particles will follow, otherwise the global best particle does not find a better place, it will not move and the global worst particle will move to this position.

The merits of ELS have been studied by its introducers. After using this strategy, the proposed algorithms' performances are improved on most multimodal functions, but the performance on unimodal functions will be slightly worse than the ones without ELS because the ELS actually add noise to the particle swarm.

In the following experiment, we will implement ELS on AIPSO and AHPSO to see whether the algorithm can be enhanced. For simplicity, the ELS will work as follows:

```
While (f<= 0.05 in 3 continuous generations)
  If (f<= 0.05in the next generation)
      The ELS will be applied.
```

The new algorithms are run on f_1, f_2,f_4, f_5 for 30 trials. The mean solution gained by each algorithm and the mean FEs (function evaluations) to reach an acceptable solution are shown in Table 5. For comparison, the AIPSO and AHPSO's testing result without ELS are also listed. So we can see that the uses of ELS and compare APSO, AIPSO and AHPSO in a relatively equitable situation.

Table 5. Result of the algorithms with&without elitist learning strategy

Function		APSO (has ELS)	AIPSO (no ELS)	AIPSO (with ELS)	AHPSO (no ELS)	AHPSO (with ELS)
f_1	Mean	1.45e-150	**1.16e-159**	1.46e-150	2.67e-159	1.15e-150
	FEs	7099	7063	8544	**6937**	7332
f_2	Mean	5.13e-84	**7.07e-89**	7.56e-84	2.19e-86	2.31e-84
	FEs	7925	7921	8162	**7343**	7911
f_4	Mean	**3.49**	11.64	5.17	9.14	3.08
	FEs	**5334**	8150	5488	5360	5396
f5	Mean	-12596.5	-7326.27	**-11261.7**	-7348.93	**-12596.5**
	FEs	7568	---	10653	---	**6911**

After utilizing the ELS approach to AIPSO and AHPSO, their performance on multimodal function *f4* and *f5* are obviously enhanced but brought down on unimodal function because ELS adds noise to the particles. Especially on *f5*, without ELS, the AIPSO and AHPSO cannot find an acceptable solution, but with ELS,AISPO perform much better than before and the AHPSO can find the global best solution. By comparing the search speed, we can find that the AIPSO and AHSPO with ELS cost much less FEs than before and become faster than APSO.

In summary, with ELS, the AIPSO and AHPSO can outperform the APSO, which means that the velocity-based inertia weight and hybrid position & velocity inertia weight can enhance the PSO, the performance of AIPSO and AHPSO are better than APSO on most problems. These results meet our expectation that by utilizing velocity information, the PSO can be improved.

3.4 Monitoring of the Inertia Weight

To monitor the behavior of the new inertia weight, we run AHPSO on Quadric function, and draw the time-varying *f* and inertia weight in figure 1.

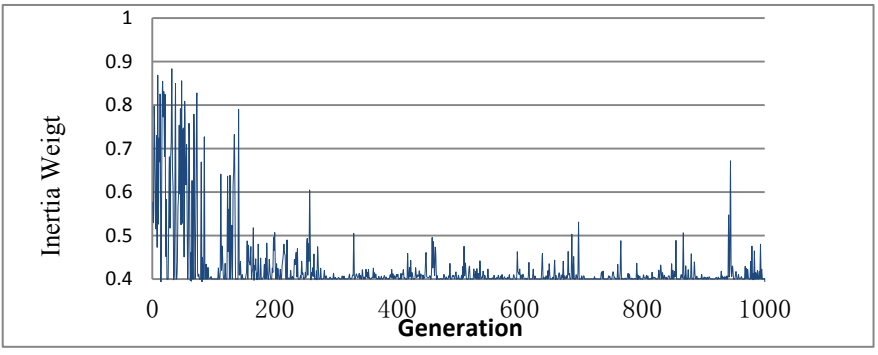


Fig. 1. The time-varyinginertia weighton Quadric function

It can be seen from Figure.1 that during 1000 evolutionary generations, the *w* is large at the early stage, then decreases rapidly, after that PSO is seen to jump out, which lead to a large value of *w*, after that *w* drops until another convergence state. The behavior of the inertia weight suggests that the PSO with this inertia weight can find a potential optimal solution because the value of *w* reflects the evolutionary state and the PSO has the potential to jump out of local optimum, that is, the inertia weight has an abrupt rise after the convergence.

5 Conclusion

In this paper, we proposed two methods to adaptively adjust the inertia weight by using velocity information. Borrowed the idea from APSO, first the velocity information is utilized instead of the position information in APSO. Another method utilized both the position and velocity information to adjust the inertia weight. Experimental tests on seven benchmark functions were conducted to compare the proposed AIPSO and AHPSO with several existing PSOs including APSO. The experimental results show that the proposed methods exceed the other PSOs on unimodal functions. For multimodal functions, AIPSO and AHPSO with elitist learning strategy (ELS) in general has better performance than APSO, which is reasonable because the APSO utilizes position information which reflects the potential solutions themselves directly while the AIPSO (AHPSO) utilizes the velocity information which reflects the changing tendency of the potential solutions but not (only) the solutions themselves. More importantly, the simulation results illustrate that the use of velocity information can effectively improve the PSO's performance, which is the purpose of this paper. In our future work, we are considering to utilize position, velocity, and cognitive diversity for inertia weight adaptation.

References

[1] Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proc. IEEE Int. Conf. Neural Networks, vol. 4, pp. 1942–1948 (1995)
[2] Eberhart, R.C., Kenedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proc. 6th Int. Symp. Micro Machine and Human Science, pp. 39–43 (1995)

[3] Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: Proc. IEEE World Congr. Comput. Intell., pp. 69–73 (1998)
[4] Shi, Y., Eberhart, R.C.: Empirical study of particle swarm optimization. In: Proc. IEEE Congr. Evol. Comput, pp. 1945–1950 (1999)
[5] Clerc, M., Kennedy, J.: The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. 6(2), 58–73 (2002)
[6] Zhan, Z., Zhang, J., Li, Y., Chung, H.S.: Adaptive particle swarm optimization. IEEE Trans. Syst., Man, Cybern. Part-B. Appl. Rev. 39(6), 1362–1381 (2009)
[7] Yao, X., Liu, Y., Lin, G.M.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. 3(2), 82–102 (1999)

Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization

Wudai Liao, Junyan Wang, and Jiangfeng Wang

School of Electrical and Information
Zhongyuan University of Technology
Zhongyuan West Road 41, Zhengzhou, Henan Province, China
wdliao@zzti.edu.cn, wangjunyan21@sina.com, jizewjf@163.com

Abstract. A nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization (NDWPSO) was presented to solve the problem that it easily stuck at a local minimum point and its convergence speed is slow, when the linear decreasing inertia weight PSO (LDWPSO) adapt to the complex nonlinear optimization process. The rate of particle evolution changing was introduced in this new algorithm and the inertia weight was formulated as a function of this factor according to its impact on the search performance of the swarm. In each iteration process, the weight was changed dynamically based on the current rate of evolutionary changing value, which provides the algorithm with effective dynamic adaptability. The algorithm of LDWPSO and NDWPSO were tested with three benchmark functions. The experiments show that the convergence speed of NDWPSO is significantly superior to LDWPSO, and the convergence accuracy is improved.

Keywords: Particle Swarm Optimization (PSO), Inertia weight, Rate of particle evolution changing, Adaptability.

1 Introduction

Particle Swarm Optimization (referred to as PSO) algorithm is a global search strategy based on population evolution of computing technology, first proposed by James Kennedy and Russell Eberhart in 1995 (see Ref[1,2]). It is through group cooperation and competition in the particles produced by the guidance of swarm intelligence optimization search, while taking advantage of its unique track memory function so that it can dynamically adjust the current search conditions to search strategy. PSO algorithm has fast convergence, set few parameters and simple operation,very suitable for optimization problems (see Ref[3,4]). But its drawback is that the process of PSO algorithm in the optimization and prone to premature and poor performance of global convergence.

Based on the PSO algorithm's insufficiency, this article has proposed NDW-PSO, it is by changing the existing algorithms inertia weight update formula, adjusted to achieve the convergence speed and jump out of local minima. Simulation results show that the modified particle swarm algorithm stability and convergence than the existing PSO significantly improved.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 80–85, 2011.
© Springer-Verlag Berlin Heidelberg 2011

2 Particle Swarm Optimization Algorithm

In the basic particle swarm algorithm, each individual is regarded as in the optimization space not to have the quality, not to have the volume particle, particle position represents the potential solution space for optimization. Each iteration, the velocity of the particles by the historical movement of individuals and groups influence state information, and to individuals and groups best position to adjust current direction and the velocity of movement of particles.

In continuous space coordinates, the mathematical description of PSO is as follows: Supposes the population size is N, the particle i in the N-dimensional space coordinates can be expressed as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, Its speed is $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD}), i = 1, 2, \cdots, M$.Records the optimal location which the particle i searches is $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$, also known as p_{best}. Search the optimal location of particles is $P_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$, also known as g_{best}. Thus, the particle i according to the formula (1) and formula (2) adjusting the speed and position of the individual.

$$v_{id}(t+1) = wv_{id}(t) + c_1 r_1(p_{id} - x_{id}(t)) + c_2 r_2(p_{gd} - x_{id}(t)) \tag{1}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{2}$$

Where, w non-negative constant, called the inertia factor; c_1 and c_2 are called learning factors, usually $c_1 = c_2 = 2.0$; r_1 and r_2 are two random numbers between 0 and 1 values; t for the current iteration.

3 NDW-PSO

Inertia weight w on the performance of particle swarm optimization has a great influence(see Ref[5]), is used to control the preceding generation of speed to the current generation of speed influence, the larger w can strengthen the global PSO search ability, the smaller w enhance the local search capacity. The experiment was later found that the dynamic weight value can be better than the result of optimization, therefore Y. Shi and R. Eberhart proposed the LDWPSO algorithm, so that the value of w increased gradually with the number of iterations is linear reduced. The linear expression is

$$w = w_{final} + \frac{Iter_{max} - Iter}{Iter_{max}} \times (w_{initial} - w_{final}) \tag{3}$$

$w_{initial}$ expresses the initial weight, w_{final} expresses the final weight, $w_{initial}$ and w_{final} recommended to take 0.9 and 0.4, respectively.The change of w associated only with the number of iterations, not more better adapt to changes in characteristics of those with complex nonlinear optimization problem. To overcome the shortcomings above algorithm, the paper introduces a nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization (NDW-PSO).

Below we give the concept of the rate of particle evolution changing.

Definition 1. *The current rate of particle evolution changing is defined as formula (4):*

$$k = \frac{P_{gbest}(T-1) - P_{gbest}(T)}{P_{gbest}(T-1)} \tag{4}$$

We introduced the rate of particle evolution changing to make the improvement to the formula (3), such as formula (5) shows:

$$w = w_{final} + \frac{Iter_{max} - Iter}{Iter_{max}} \times (w_{initial} - w_{final}) \times k \tag{5}$$

In the particle swarm algorithm, global optimum value is determined by the optimal value of the individual, and in the iteration process the value of the current global optimum is always better than or at least equal to the previous iteration of the global optimal value. In view of the fact, this has introduced the rate of particle evolution changing. $P_{gbest}(T-1)$ and $P_{gbest}(T)$ show the preceding time iterates global optimum value and the current global optimum value respectively.

The rate of particle evolution changing had considered the particle beforehand running status, responded the reaction speed of the particle group in the degree of evolution. When the fast evolution of large, the algorithm can continue in a larger search space, when is small may reduce w causes the particle to search in the small scope, thus more quickly find the optimal value. k value of early large, fast speed of evolution, and vice versa. When after several iterations, show that the algorithm is stagnant or has found the optimal value. Can be drawn, w decreases as k decreases.

The above analysis shows that the introduction of the rate of particle evolution changing, making the particle swarm algorithm to modify its inertia factor, not only to consider the role of the number of iterations, and the group best position to consider the impact, so as to better balance the global search ability of particle swarm and local search capabilities to avoid falling into local optimum, to speed up the convergence speed.

4 Simulation Results

To test the validity of the new method, this paper carries on the simulation with the LDWPSO and NDWPSO algorithm to the three classical Benchmark functions. All of the following functions have the same global minimum value 0. Where function 1 is a single peak function, the other for the Multi-peak function.

Function 1 Rosenbrock function:

$$f_1(x) = \sum_{i=1}^{n-1}((100x_{i+1} - x_i^2)^2 + (x_i - 1)^2), -50 < x_i < 50$$

Function 2 Rastrigrin function:

$$f_2(x) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10), -50 < x_i < 50$$

Function 3 Griewank function:

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1, -600 < x_i < 600$$

Parameter settings: Size of the swarm is 20; Maximum number of iterations is 1000; Dimension of the problem is 10; $w_{initial} = 0.9$; $w_{final} = 0.4$; $c_1 = c_2 = 2.0$ The target value of function set to 1.0e-5.

Figure 1 - Figure 3 shows that the use of LDWPSO and NDWPSO algorithms to simulate these functions results.

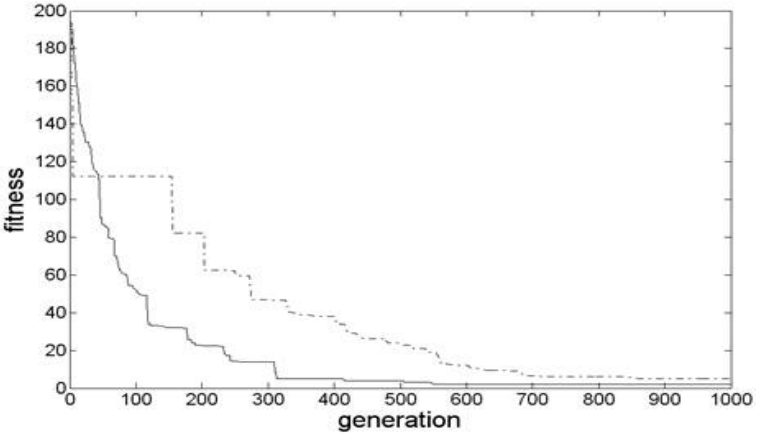


Fig. 1. Rosenbrock function iterative process

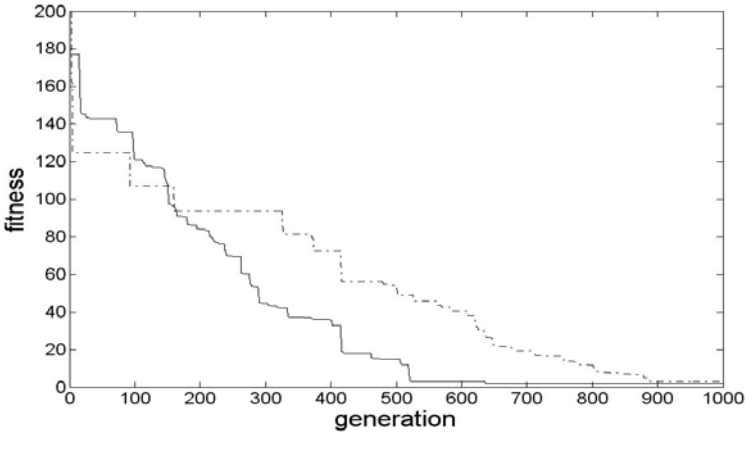


Fig. 2. Rastrigrin function iterative process

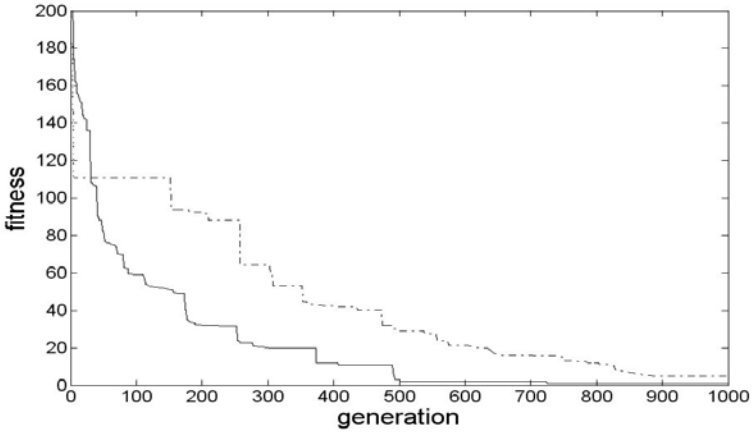


Fig. 3. Griewank function iterative process

Figure 1 - Figure 3 show that NDWPSO algorithm than LDWPSO algorithm has significantly improved in the search ability and convergence speed of the optimal values.

5 Concluding Remarks

In this paper, based on the linear decreasing inertia weight PSO (LDWPSO) analysis and found LDWPSO adapt to the complex nonlinear optimization in the search process, there are problems like it easily stuck at a local minimum point and its convergence speed is slow, in order to overcome this, a nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization (NDWPSO) has been provided. Through introduction the rate of particle evolution changing concept, change inertia factor renewal formula, the algorithm achieved global and local search performance of the balance, and improved LDWPSO algorithm easily stuck at a local minimum point. To show effectiveness of this method, the simulations of three benchmark examples are carried out by the proposed method, as a result, NDWPSO algorithm in the stability and convergence better than LDWPSO algorithm, and the algorithm is easy to combine with other algorithms.

Acknowledgement

This work was supported by Natural Science Funds of Ministry of Education of Henan Province (2011A120009).

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE, Piscataway (1995)

2. Kennedy, J.: The Particle Swarm. In: Preceedings of Evolutionary Computation, Social Adaptation of Knowledge, pp. 303–308. IEEE, Indianapolis (1997)
3. Su, C.-T., Wong, J.-T.: Designing MIMO controller by neuro-traveling particle swarm optimizer approach. Expert System with Applications (S0957-4174) 32(3), 848–855 (2007)
4. Kim, T.-H., Maruta, I., Sugie, T.: Robust PID controller tuning based on the constrained particle swarm optimization. Automatica (S0005-1098) 44(4), 1104–1110 (2008)
5. Zhang, L., Yu, H.: Analysis and Improvement of Particle Swarm Optimization. Information and Control 33(5), 513–517 (2004)

An Adaptive Tribe-Particle Swarm Optimization

Yong Duan Song[1], Lu Zhang[1], and Peng Han[2]

[1] Intelligent Systems and Renewable Energy Research Center,
Beijing Jiaotong University
[2] Chongqing Institute of Science and Technology
songyd_ncat@yahoo.com

Abstract. This paper talks about the problems in particle swarm optimization (PSO), including local optimum and difficulty in improving solution accuracy by fine tuning. We presents a new variation of Adaptive Tribe-PSO model where nonlinear updating of inertia weight and a particle's fitness with Tribe-PSO model are combined to improve the speed of convergence as well as fine tune the search in the multidimensional space. The method proved to be a powerful global optimization algorithm.

Keywords: adaptive weight, tribe particle swarm optimization, local optimum, accuracy.

1 Introduction

Particle swarm optimization (PSO) algorithm is a new intelligent optimization algorithm [1]. Basic PSO can easily get trapped in the local optima when solving multimodal problems [2]. Accelerating convergence speed and avoiding the local optima have become the two most appealing goals in PSO research [3].

Tribe-PSO in [4] has shown some encouragement. It focuses on avoiding the local optima, but brings in a slower convergence as a result. One of the reasons is the computational burden of breeding and subpopulation [5]. Another is the fixed inertia weight. Proper coordination between global and local search is critical for algorithm finally converging to global optimal solution [6]. The time-varying controlling strategies proposed for the PSO parameters so far are according to either linear [7], [8] or nonlinear rules. Some use a self-adaptive method by optimizing parameters together with the position during run time [9], [10]. Some strategies use a fuzzy system with fitness feedback to adjust the parameters [11]. But they are under the risk of inappropriately adjusting the parameters without information that reflects evolutionary state utilized. An adaptive inertia weight developed in [12] has been proved to reflect the population and fitness diversity well. In this paper, we present a new variation of Adaptive Tribe-PSO model where nonlinear updating of inertia weight and a particle's fitness with a Tribe-PSO model are combined to improve the speed of convergence as well as fine tune the search. Benchmarking tests indicate that the proposed method is able to achieve satisfactory results.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 86–92, 2011.
© Springer-Verlag Berlin Heidelberg 2011

2 Basic PSO Algorithm

In Particle swarm optimization (PSO), particles find the optimal solution by updating the velocity and position with the following equations:

$$V_{i,d}^{t+1} = wV_{i,d}^{t} + c_1 rand_1\left(pBest_{i,d}^{t} - x_{i,d}^{t}\right) + c_2 rand_2\left(gBest_{d}^{t} - x_{i,d}^{t}\right) \tag{2.1}$$

$$x_{i,d}^{t+1} = x_{i,d}^{t} + V_{i,d}^{t+1} \tag{2.2}$$

where c_1 and c_2 are the learning factor set to 2 to give the two stochastic factors $rand_1\left(pbest_{i,d}^{t} - x_{i,d}^{t}\right)$ and $rand_2\left(gbest_{d}^{t} - x_{i,d}^{t}\right)$ a mean of 1, so that agents would "overfly" the target about half the time. Here $rand_1$ and $rand_2$ are in the range of [0, 1]. $V_{i,d}^{t}$ and $x_{i,d}^{t}$ are the velocity and position of ith particle in dth dimension till tth iteration, respectively. $gbest_{d}^{t}$ and $pbest_{i,d}^{t}$ are the global best and the personal best, respectively. As all particles are affected by the same social optimum, they will all approach to it. The swarm lost particles diversity and even leads to premature.

3 Tribe-PSO

The two-layered structure of Tribe-PSO is depicted in Fig. 1. Tribe-PSO divides particles into two layers and procedure of optimization is divided into three phases.

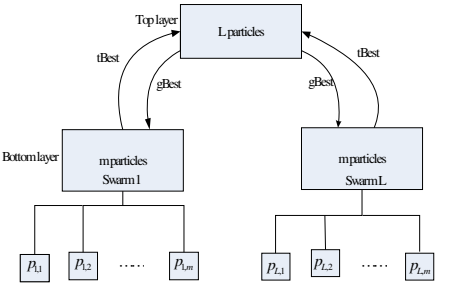


Fig. 1. The structure of Tribe-PSO

Assume that there are $l \times m$ particles. They are firstly divided into l tribes with each tribe m particles. The convergence procedure of Tribe-PSO consists of three phases:

1) Isolated phase
In the first phase, the tribes work as l independent basic PSO models. Every tribe produces a best particle *tBest*. The velocity function is defined as followed:

$$V_{i,d}^{t+1} = wV_{i,d}^{t} + c_1 rand_1\left(pBest_{i,d}^{t} - x_{i,d}^{t}\right) + c_2 rand_2\left(tBest_{d}^{t} - x_{i,d}^{t}\right) \tag{3.1}$$

2) Communing phase

In the second phase, Tribe-PSO works in the standard two-layered model: all tribes form the basic layer and all *tBests* form the upper layer. For Tribe members, the velocity function is the same as they have in the first phase:

$$V_{i,d}^{t+1} = wV_{i,d}^{t} + c_1 rand_1\left(pBest_{i,d}^{t} - x_{i,d}^{t}\right) + c_2 rand_2\left(tBest_{d}^{t} - x_{i,d}^{t}\right) \tag{3.2}$$

For *tBest* and *gBest* particles, the velocity function is defined as followed:

$$V_{i,d}^{t+1} = wV_{i,d}^{t} + c_1 rand_1\left(pBest_{i,d}^{t} - x_{i,d}^{t}\right) + c_2 rand_2\left(gBest_{d}^{t} - x_{i,d}^{t}\right) \tag{3.3}$$

3) United phase

In the last phase, all the tribes are united into one group. The velocity function for all the particles becomes the original one in the basic PSO.

$$V_{i,d}^{t+1} = wV_{i,d}^{t} + c_1 rand_1\left(pbest_{i,d}^{t} - x_{i,d}^{t}\right) + c_2 rand_2\left(gbest_{d}^{t} - x_{i,d}^{t}\right) \tag{3.4}$$

4 Adaptive Tribe-Particle Swarm Optimization

4.1 Adaptive Inertia Weight

Here we compute an "evolutionary factor" *s* based on the evolutionary state as defined by

$$s = \frac{f - f_{\min}}{f_{avg} - f_{\min}} \in [0,1] \tag{4.1}$$

f denotes the particle's fitness at present and f_{avg} the average fitness, f_{min} the minimum fitness. The dynamics of *s* has been shown in figure 2.

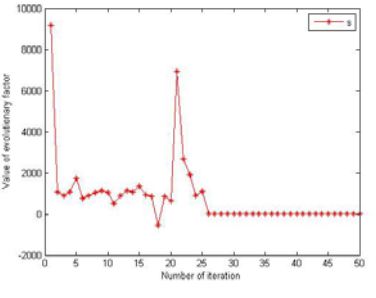


Fig. 2. *s* from time-varying Griewank function f_2

Based on the dynamics of *s*, we can allow *w* to follow the evolutionary states using a mapping $\omega(s):\Re^+ \mapsto \Re^+$:

$$w(s)=\begin{cases} w_{\min}+(w_{\max}-w_{\min})*s & f<f_{avg} \\ w_{\max} & f\geq f_{avg} \end{cases} \quad w(s)\in[0.4,0.9] \quad \forall s\in[0,1] \tag{4.2}$$

It can be seen that *w* is monotonic with *s*. Thus *w* will adapt to the search environment characterized by *s*.

$f > f_{avg}$: The largest *w* makes it fly to the optimum and accelerates convergence speed.
$f < f_{avg}$: The small *w* helps to protect excellent particles and limits their research area.
$f = f_{avg}$: The largest *w* to jump out of local optimum or enlarge search area and keep the diversity of particles.

4.2 Adaptive Tribe-PSO

Based on the previous analysis and discussion, we can establish adaptive Tribe-PSO as described by the following flow chart:

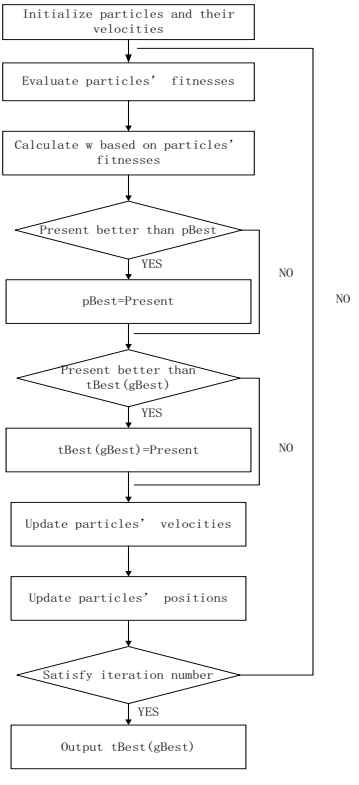


Fig. 3. The Flow Chart of the proposed Adaptive Tribe-PSO

The proposed Adaptive Tribe-PSO mainly constitutes the following steps.

- Step 1 Initialization.ïSet parameters. Initialize particle position and velocity.
- Step 2 Calculate fitness. Based on the fitness, choose *tBest, gBest* or *pBest*.
- Step 3 Update. Update velocity and position using formula (3.1)—(3.4), (2.2).
- Step 4 Output optimum. In the first two phases, algorithm goes to next phase with the best values. If that is in the third phase, output the optimum.

5 Experimental Setup and Simulation

5.1 Benchmark Functions

Simulation tests are conducted on two benchmark functions: Spherical and Griewank. f_1: Spherical function:

$$f_1(x) = \sum_{i=1}^{n} x_i^2 \qquad -100 \le x_i \le 100 \tag{5.1}$$

f_2: Griewank function:

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1 \qquad -100 \le x_i \le 100 \tag{5.2}$$

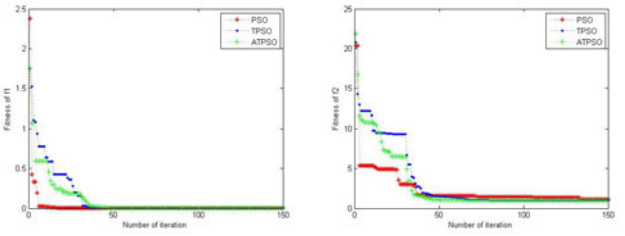
5.2 Parameter Settings

Firstly, strict benchmarking tests are carried out for PSO, Tribe-PSO and Adaptive Tribe-PSO. All the tests start from the same random initiation: the dimension number is set to 10; number of particles is set to 200; number of iterations is set to 150; w is initialized to 0.7, and c_1 and c_2 to 2.0. w_{max} and w_{min} is set to 0.9 and 0.6, respectively. The particles are divided to 20 tribes with each tribe 10 particles. The partition of phases is set to 0.2, 0.3 and 0.5, respectively. Secondly, for Adaptive Tribe-PSO, the partition of phases and ratios of tribe numbers to tribe size are initialized differently.

5.3 Results

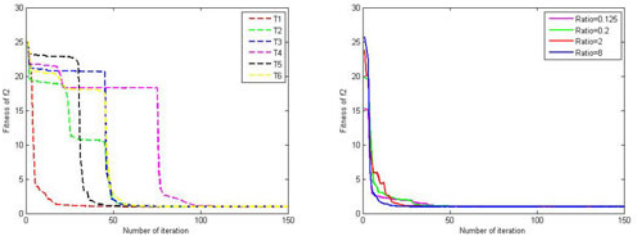
During the isolated phase, basic PSO has the smallest optimal value. In communing phase, two modified PSOs have better optimized value. And Adaptive Tribe-PSO converges earlier than Tribe-PSO. Two modified PSOs get better results than basic PSO at last. In Tribe-PSO the attraction from the best particle has been limited by the two-layered structure. In Adaptive Tribe-PSO, this problem is resolved well. Particles can adjust velocity intelligently and smash the restriction of two-layered structure.

In Fig. 5 (a), T1-T6 represent the trajectory with partitions of phases (0.02, 0.3, 0.68), (0.3, 0.02, 0.68), (0.3, 0.68, 0.02), (0.5, 0.3, 0.2), (0.2, 0.3, 0.5) and (0.3, 0.5, 0.2), respectively. T1, T2, T5 get more accurate optimum where the third phase occupies more. When the first phase occupies too much portion, it has worse convergence result. In Fig. 5 (b), it indicates that the upper layer with more tribes results in better convergence as long as each tribe has enough particle members.



(a) (b)

Fig. 4. Average convergence curves of Spherical's function (a) and Griewank's function (b)



(a) (b)

Fig. 5. (a) Average convergence curves of Griewank's function with different partition of phases (b) Average convergence curves of Griewank's function with different number of tribes

6 Conclusion

In this paper, the problem of premature and local optimum in PSO is investigated and Tribe-PSO has been extended to Adaptive Tribe-PSO by introducing dynamically and adaptively weights updating. As shown in the tests, the proposed approach can achieve better accuracy. Both the partition of phases and number of tribes can be adjusted according to the practical problem. Further more, the adaptive weight updating employed in the algorithm can effectively prevent local optimum.

References

[1] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)

[2] Liang, J.J., Qin, A.K., Suganthan, P.N., Bizataskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. 10(3), 281–295

[3] Zhan, Z.-H., Zhang, J., Li, Y., Chung, H.S.-H.: Adaptive Particle Swarm Optimization. IEEE Trans. Syst., Man, Cybern. C 39(6), 1362–1381 (2009)

[4] Chen, K., Li, T.H., Cao, T.C.: Tribe-PSO: A novel global optimization algorithm and its application in molecular docking. Chemometrics and Intelligent Laboratory Systems 82(1-2), 248–259 (2006)
[5] Chatterjee, A., Siarry, P.: Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Comput. Oper. Res. 33(3), 859–871 (2004)
[6] Shi, Y., Eberhart, R.C.: Fuzzy adaptive particle swarm optimization. In: Proc. IEEE Congr. Evol. Comput., vol. 1, pp. 101–106 (2001)
[7] Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proc. IEEE World Congr. Comput. Intell., pp. 69–73 (1998)
[8] Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evol. Comput. 8(3), 240–255 (2004)
[9] Tripathi, P.K., Bandyopadhyay, S., Pal, S.K.: Adaptive multi-objective particle swarm optimization algorithm. In: Proc. IEEE Congr. Evol. Comput., Singapore, pp. 2281–2288 (2007)
[10] Ratnaweera, A., Halgamuge, S., Watson, H.: Particle swarm optimization with self-adaptive acceleration coefficients. In: Proc. 1st Int. Conf. Fuzzy Syst. Knowl. Discovery, pp. 264–268 (2003)
[11] Yamaguchi, T., Yasuda, K.: Adaptive particle swarm optimization: Self-coordinating mechanism with updating information. In: Proc. IEEE Int. Conf. Syst., Man, Cybern., Taipei, Taiwan, pp. 2303–2308 (October 2006)
[12] Gong, C., Wang, Z.L.: Optimization calculation based on MATLAB, pp. 283–285. Publish House of Electronics Industry, Beijing (2009)

A Novel Hybrid Binary PSO Algorithm

Muhammd Ilyas Menhas, MinRui Fei, Ling Wang, and Xiping Fu

Shanghai Key Laboratory of Power Station Automation Technology,
School of Mechatronics Engineering and Automation, Shanghai University,
Shanghai 200072, China
ilyasminhas75@yahoo.com

Abstract. The continuous PSO algorithm has been widely researched and also applied as an intelligent computational technique to solve problems requiring iterative solutions based on some predefined objective function. However, the research on binary version of PSO (DBPSO) is still underway. The major research concerns are to accelerate the convergence speed retaining the search ability and reliability of the algorithm. To achieve this, a novel hybrid binary particle swarm optimization (HBPSO) algorithm is proposed in this paper. It combines the PSO's concept and GA. In the existing standard binary PSO (DBPSO) two new operators such as crossover and mutation are incorporated to accelerate the convergence speed and to avoid possible stuck in local optimum thereby maintaining population diversity. The proposed HBPSO algorithm has been studied on 6 bench mark optimization problems. The experimental results such as minimum fitness, mean fitness, and variance of fitness over 50 consecutive trials on each objective function indicate that the HBPSO algorithm consistently outperforms the DBPSO and its variants in terms of convergence speed and search accuracy on a bulk of bench mark problems.

Keywords: Particle swarm optimization, discrete binary PSO, optimization.

1 Introduction

Particle swarm optimization algorithm (PSO) is a population based stochastic search algorithm developed by Kennedy and Eberhart [1] in 1995. It mimics swarms behavior in performing their tasks like bird flocks and fishes to discover an optimal solution based on an objective function. In contrast to the traditional optimization methods based on pure mathematics, the PSO employs a probabilistic search procedure and iteratively determines solutions to an optimization problem. The PSO comprises two main variants namely the continuous PSO [1] and the binary PSO (DBPSO) [2] algorithms. The continuous PSO mainly focuses on problems requiring real numbers while the binary PSO deals with problems that need binary encoding like knapsack and similar decision oriented problems.

The (DBPSO) [2] preserves the fundamental concept of the PSO algorithm except that here a population of candidate solutions consists of binary strings each representing a particle's position vector. The initial bit streams for the whole population are randomly generated subject to arbitrary velocities. The fitness of each individual of the population in minimizing or maximizing a predefined objective function $F(x)$ is

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 93–100, 2011.
© Springer-Verlag Berlin Heidelberg 2011

evaluated and remembered. The individuals are sorted according to their fitness values. An individual with the best fitness value is identified as a local best from the current population of candidate solutions. Then, an iterative process begins to alter the bit streams with some probability to facilitate further movements in the search space. Now, updating of bits is accomplished with renew of the pseudo probability $\boldsymbol{V}_{\mathrm{ij}}$ as follows:

$$\overset{K+1}{V_{ij}} = w.\overset{K}{V_{ij}} + r_1.c_1.(\overset{K}{p_{ij}} - \overset{K}{x_{ij}}) + r_2.c_2.(\overset{K}{g_{1j}} - \overset{K}{x_{ij}}) \tag{1}$$

where $\overset{K}{V_{ij}}$ is the pseudo probability; x_{ij} is a bit of candidate solution; i is the index of the candidate solution and j is the index of dimension; c_1, c_2 are two acceleration coefficients; r_1, r_2 are two random numbers; w is the inertia weight; $\overset{K}{p_{ij}}$ is the personal best history at iteration K; $\overset{K}{g_{1j}}$ is the current local best at iteration K. The pseudo probability $\overset{K}{V_{ij}}$ is constrained in the actual probability interval $S_{ij} \in \{0,1\}$ using a sigmoid mapping function as below:

$$S_{ij} = \frac{1}{(1+\exp(-V_{ij}))} \tag{2}$$

Now, a random number $r_{ij} \in \{0,1\}$ is generated and compared with $S_{ij} \in \{0,1\}$ to determine actual bit as

$$x_{ij} = \begin{cases} 1 & r_{ij} <= s_{ij} \\ 0 & else \end{cases} \tag{3}$$

The procedure of generating new candidate solutions, fitness evaluation, renew of personal best, and local best, is continued until the global best solution is ascertained or a predefined termination criterion is reached.

In fact, both PSO methods were developed for diverse work domains using the same underlying principles but the binary PSO has extensive applications as the sequence of binary bits can be transformed according to the requirements of any problem space. Also the binary coded PSO can overcome the problems such as trapping into local optima's because of the inherited advantage of bit flip mechanism however the algorithm might be computationally expensive due to extra computations involved in the process of number conversion and scaling.

As the binary PSO is suitable for a wide range of applications of diverse nature and comprises simple mathematics, it has attracted many researchers for its further development. Shen [3] proposed a modified version of binary PSO (MBPSO) by suggesting some modifications to the original version of the binary PSO. Though the modifications proposed by Shen have improved performance of the algorithm in terms of its convergence speed but the MBPSO algorithm traps in local optimum and possesses a high fitness variance that restricts the applications requiring consistency.

Ling Wang [4] further extended the MBPSO by introducing two new operators called mutation and dissipation, concepts of natural evolutionary theory, and developed a mutation-dissipation based binary PSO (MDBPSO). Alireza [5] proposed a Boolean PSO algorithm by introducing the concepts of boolean algebra in binary PSO algorithm and designed a dual-band dual-polarized planar antenna using the proposed method. Khansesar [6] proposed a novel binary PSO by redefining the concept of velocity vector in binary PSO. Arezoo Moiri [7] developed a modified binary PSO algorithm by adapting a new velocity calculation strategy to improve the accuracy and the convergence speed. Jeong [8] used a hybridized approach to combine the concept of quantum computing and binary PSO and developed a quantum-inspired binary PSO (QBPSO). Further, essential binary PSO (EBPSO) and probability based binary PSO (PBPSO) have also been proposed [9-10].

With the ongoing progress in research, the binary PSO has been successfully applied in a wide range of optimization problems such as in process control, structural topology optimization, chaotic map feature selection, optimal placement of PMU's in power network, knapsack problem, management, scheduling, parameter identification, wireless sensor networks, resource allocation, neural network training and face recognition problem. The application results witness the significance of binary PSO's; nevertheless there is still room to improve the method to achieve better results.

2 The Hybrid Binary PSO (HBPSO) Algorithm

Despite the standard binary PSO (DBPSO) has been applied in many real world optimization problems but its performance is not satisfactory. Firstly, the algorithm is computationally expensive due to slow convergence. Moreover, the core concept of the PSO algorithm which governs the movements of particles to their historical best positions and toward the best particle in the group is not completely present in binary PSO. Although, it employs the PSO parameters such as the acceleration factors and the inertia weight in the course of updating the pseudo probability, but later it is only a probability constrained by a sigmoid mapping function in the real probability interval to determine a bit's state. It cannot aid algorithm to avoid unnecessary march in the search space and in attaining convergence, consequently, it increases the required number of iterations for the algorithm to converge to an optimum solution.

Many applications cannot afford large number of fitness evaluations, because it may take long time to evaluate a candidate solution due to the complexity of objective function and other constraints. In order to establish speedy convergence, reduce wandering, and avoid possible trap in local optimum a hybrid approach is proposed. It combines concepts of the PSO and GA [11]. The proposed hybrid strategy comprises a crossover and a mutation operator. An intelligent stochastic crossover operation is performed to wholly realize the cognitive and social concepts of the PSO and then a mutation operator is added as a remedy against possible stuck in local optimum as well as to maintain population diversity. The hybridization has significantly improved performance of the proposed HBPSO in terms of convergence speed, consistency, and accuracy of the solution.

2.1 Crossover and Mutation

Mutation and crossover are fundamental concepts employed in genetic algorithm (GA) derived from natural evolution theory. These operators mimic the biological evolutionary process.

(1) Crossover

In the GA, a crossover operation is performed to generate new candidate solutions for next generation from an existing population of solutions by performing recombination. There are numerous ways to establish the crossover operation. Like GA, in HBPSO an intelligent stochastic crossover operation is introduced to indemnify the essence of the particle swarm optimization in binary domain. Here, a random number $r_{ij} \in \{0,1\}$ is generated to determine the actual state of each bit of every candidate solution from the population of current solutions as in eq.4.

$$x_{ij} = \begin{cases} x_{ij} & \text{if } 0 \le r_{ij} \le \alpha \\ p_{ij} & \text{if } \alpha < r_{ij} \le 2\alpha \\ g_{ij} & \text{if } 2\alpha < r_{ij} \le 1 \end{cases} \tag{4}$$

In eq.4, α represents a predefined percentage ratio of crossover probability.

In HBPSO, $\alpha = 33.33\%$ i.e. for each bit, x_{ij} of a candidate solution, the possibility to remain in its state as it was determined by equation (3) is 33.33% and 66.67% chances to be replaced by a bit from its previous best history and the current best particle at any iteration. Earlier, a similar approach has been applied in the MBPSO algorithm [3].However; the MBPSO is quite different from the proposed HBPSO algorithm. Because there an initial population of candidate solutions is created with a static probability and then a random number $r_{ij} \in \{0,1\}$ is generated to realize an identical approach called update strategy but in fact it results in a very rapid convergence to a local optimum, and in many cases very far from real optimum solution, without exploring search space well. To investigate search space well again, it is necessary to push particles ahead to discover new points with some reasonable velocity. In contrast, the proposed HBPSO preserves the DBPSO's strategy of generating candidate solutions with a dynamic probability during all iterations as in equation (3) and then performs a stochastic operation analogous to the uniform crossover in GA as in equation (4) to accelerate convergence. The HBPSO has less chances of premature convergence as particles will consistently move further to find new points.

(2) Mutation

Mutation is another commonly used operator in genetic algorithm (GA) to maintain population diversity. In HBPSO, the mutation operator is added to achieve two goals. Firstly, it can maintain diversity of the swarm; in addition, it can facilitate escape when algorithm has converged to any local optimum. A random number $r_{ij} \in \{0,1\}$ is generated and compared with a predefined mutation probability $\boldsymbol{m}$ to perform mutation operation as in eq.5.

$$x_{ij} = \begin{cases} \overline{x_{ij}} & if \ r_{ij} <= m \\ x_{ij} & else \end{cases} \tag{5}$$

In the case of problems other than binary, a mutation operation on a most significant bit (MSB) of any candidate solution will facilitate exploration (global search) while mutation of a least significant bit (LSB) will aid exploitation (local search).The overall procedure of the proposed HBPSO algorithm is illustrated with the help of a flow chart as in Fig.1.

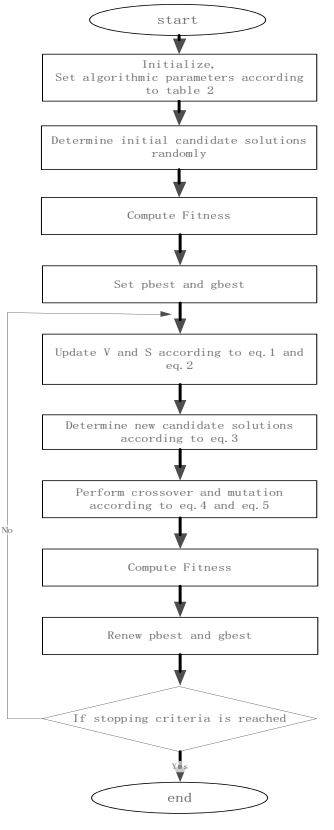


Fig. 1. Flowchart of the HBPSO algorithm

3 Experiments and Numerical Optimization Results

Experiments were performed using Matlab, Intel® Pentium® Dual-CPU T2370 @1.73GHz and 2GB of RAM. To evaluate the performance of the proposed HBPSO algorithm 6 bench mark problems were considered. The characteristics of bench mark problems are given in Table 1.

Table 1. Characteristics of Benchmark Functions

	Name	Dimension	Optimum	Type
F_1	Sphere	2	0.0(min)	unimodal
F_2	Step	2	0.0 (min)	unimodal
F_3	Rosenbrock's	2	0.0(min)	unimodal
F_4	Griewangk's Function	2	0.0(min)	unimodal
F_5	Glankwahmdee's	2	0.0(min)	unimodal
F_6	Rastrigin	2	80.70658(max)	multimodal

Table 2. Parameter settings of DBPSO, MBPSO, MDBPSO, and HBPSO algorithm

DBPSO	C1=C2=2, Vmin=-Vmax=5 Inertia weight (w)=0.8 Swarm size=30 No. of binary bits=24 Total number of iterations=500	MDBPSO	Static probability=0.5 Mutation probability=0.005 Dissipation probability=0.01 Swarm size=30 No. of binary bits=24 Total number of iterations=500
MBPSO	Static probability=0.5 Swarm size=30 No. of binary bits=24 Total number of iterations=500	HBPSO	C1=C2=2, Vmin=-Vmax=5 Inertia weight (w)=0.8 Mutation probability=0.005 Cross over probability=66.67% Swarm size=30 No. of binary bits=24 Total number of iterations=500

Table 3. Numerical Simulation results of DBPSO, MBPSO, MDPSO, HBPSO algorithm

F	F_V	DBPSO	MBPSO	MDBPSO	HBPSO	F	F_V	DBPSO	MBPSO	MDBPSO	HBPSO
F_1	B_{FV}	1.78e-9	5.72e-6	7.11e-11	7.11e-11	F_4	B_{FV}	8.37e-7	6.59e-5	9.59e-10	9.59e-10
	M_{FV}	1.15e-5	1.2808	7.11e-11	7.67e-11		M_{FV}	0.0323	4.9513	0.0797	0.0401
	V_F	4.16e-10	40.881	2.73e-51	1.62e-21		V_F	0.0052	200.0113	0.0127	0.0065
	H_F	9.76e-5	44.1662	7.11e-11	3.55e-10		H_F	0.2037	77.5925	0.4465	0.2004
F_2	B_{FV}	1.05e-7	1.82e-4	2.44e-7	3.34e-10	F_5	B_{FV}	8.48e-8	7.02e-5	5.54e-11	7.50e-12
	M_{FV}	0.2406	1.2560	0.2282	0.2250		M_{FV}	0.0035	0.9918	0.1926	0.0043
	V_F	0.0359	28.0818	0.0334	0.0312		V_F	8.97e-5	6.6664	0.4603	1.24e-4
	H_F	0.5040	37.4719	0.5000	0.5000		H_F	0.0459	15.0266	4.5524	0.0481
F_3	B_{FV}	5.12e-9	6.70e-6	1.57e-8	4.84e-13	F_6	B_{FV}	80.7066	80.71	80.7066	80.7066
	M_{FV}	2.64e-4	0.1677	0.0668	3.36e-4		M_{FV}	80.7066	80.5547	80.7057	80.7062
	V_F	9.72e-8	0.2024	0.0389	7.40e-7		V_F	2.26e-21	0.0866	2.25e-6	1.31e-6
	H_F	8.16e-4	2.9528	1.0000	0.0060		H_F	80.7066	79.8825	80.7066	80.7031

* B_{FV}=Best Fitness Value, M_{FV}=Mean Fitness Value, V_F=Variance of Fitness, H_F=Highest Fitness

The proposed HBPSO algorithm was compared with three former versions of binary PSO like DBPSO, MBPSO and MDBPSO algorithm. Due to sensitivity of algorithmic parameters, the parameters were chosen after extensive experimentations with different parameter combinations and also considering previously reported values in the literature. The parameters combinations were set for each algorithm as listed in Table 2. Experiments were repeated 50 times under identical conditions with each algorithm on each objective function. The experimental results like minimum fitness, mean fitness, variance of fitness, and maximum fitness values for each algorithm on each objective function over 50 consecutive trials were computed to measure performance index of each algorithm. The statistical data obtained from experiments is presented in Table 3.The convergence curves associated with the best results among 20 independent consecutive trials of each algorithm on each objective function are shown in the following figures (Fig.2).

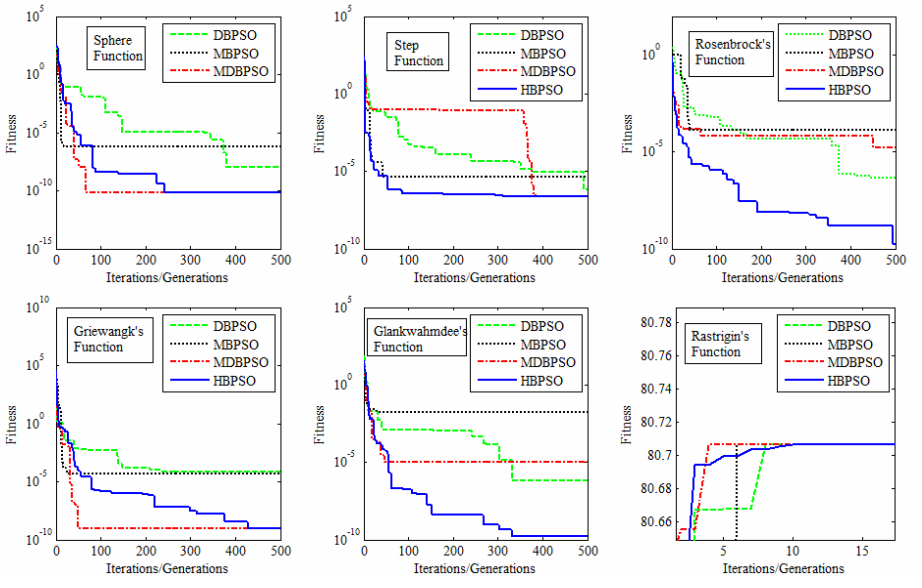


Fig. 2. Convergence curves of various algorithms on different benchmark functions

4 Conclusions and Discussion on Results

In this paper a novel HBPSO algorithm is developed by integrating concepts of PSO and GA to arrive at fast convergence and avoid excessive wander. From Table 3 it is evident that the proposed HBPSO shows the best performance index in terms of minimum fitness, mean fitness, variance of fitness in comparison to DBPSO, MBPSO and MDBPSO algorithms. The HBPSO consistently outperformed and converged closer to the optimum solution. The results should be compared in terms of mean, minimum and highest fitness because any algorithm repeatedly trapping into a local optimum also exhibits a low variance. The convergence curves (Fig.2) show that the HBPSO can reach an acceptable solution within lesser number of fitness evaluations

in most cases. Also the curves indicate that the HBPSO frequently avoids local optimum and perform search in the area of potential solutions.

Acknowledgments. This work is supported by the Projects of Shanghai Science and Technology Community (10ZR1411800, 08160705900 & 08160512100), National Natural Science Foundation of China (Grant No. 60834002 & 61074032), Research fund for the Doctoral Program of Higher Education (20103108120008) of China, Mechatronics Engineering Innovation Group project from Shanghai Education Commission, and the Graduate Innovation Fund of Shanghai University (SHUCX102218).

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948. IEEE service center, Los Alamitos (1995)
2. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108 (12-15, 1997)
3. Shen, Q., Jiang, J.H., Jiao, C.X., Shen, G.L., Yu, R.Q.: Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists. European Journal of Pharmaceutical Sciences 22(2-3), 145–152 (2004)
4. Wang, L., Wang, X.T., Fei, M.R.: An adaptive mutation-dissipation binary particle swarm optimization for multidimensional knapsack problem. International Journal of Modeling, Identification and Control 8(4), 259–269 (2009)
5. Marandi, A., Afshinmanesh, F., Shahabadi, M., Bahrami, F.: Boolean Particle Swarm Optimization and Its Application to the Design of a Dual-Band Dual-Polarized Planar Antenna. In: IEEE Congress on Evolutionary Computation, CEC 2006, pp. 3212–3218. IEEE Press, New York (2006)
6. Khanesar, M.A., Teshnehlab, M., Shoorehdeli, M.A.: A novel binary particle swarm optimization. In: Mediterranean Conference on Control & Automation, pp. 1–6 (June 2007)
7. Modiri, A., Kiasaleh, K.: Modification of real-number and binary PSO algorithms for accelerated convergence. IEEE Transactions on Antennas and Propagation 59(1), 214–224 (2011)
8. Jeong, Y.W., Park, J.B., Jang, S.H., Lee, K.Y.: A new quantum-inspired binary PSO for thermal unit commitment problems. In: 15th International Conference on Intelligent system applications, pp. 1–6 (2009)
9. Menhas, M.I., Wang, L., Pan, H., Fei, M.R.: CFBB PID Controller Tuning with Probability based Binary Particle Swarm Optimization Algorithm, LSMS, Communications in Computer and Information Science, vol. 98, pp. 44–51. Springer, Heidelberg (2010)
10. Chen, E.X., Li, J.Q., Liu, X.Y.: In Search of the Essential Binary Discrete Particle Swarm. Applied Soft Computing Journal 11(3), 3260–3269 (2010)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, Ann Arbor (1975)

PSO Algorithm with Chaos and Gene Density Mutation for Solving Nonlinear Zero-One Integer Programming Problems*

Yuelin Gao[1], Fanfan Lei[1], Huirong Li[2], and Jimin Li[1]

[1] Institute of Information & System Science, Beifang University of Nationalities, Ningxia, Yinchuan, 750021
[2] Department of Mathematics and Computer Science, Shangluo University, Shanxi, Shangluo, 726000

Abstract. By the penalty function methodwe transform zero-one nonlinear programming problems into unconstrained zero-one integer optimization problems. A particle swarm optimization algorithm with chaos and gene density mutation is given to solve unconstrained the zero-one nonlinear program problems. We use chaos to initialize populations and use the 0-1 integer operation in updating positions to produce 0-1 integer points. We use the fitness variance and gene density strategy to determine whether the population premature phenomenon or not. If it appears that we use the gene density mutation to increase the population diversity or restart and reset the population by chaos technique. Numerical simulations show that the proposed algorithm for most test functions is feasible, effective and has high precision.

Keywords: nonlinear zero-one integer programming; penalty function method; particle swarm optimization; gene density mutation.

1 Introduction

In this paper, we consider zero-one nonlinear programming problems as follows:

$$\begin{cases} \min f(\mathbf{x}), \\ s.t.\ \ g_i(\mathbf{x}) \le 0, i = 1, 2, \cdots, m, \\ \qquad h_j(\mathbf{x}) = 0, j = 1, 2, \cdots, l, \\ \qquad x_k = 0 \quad or \quad x_k = 1, k = 1, 2, \cdots, n. \end{cases} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, $g_i(\mathbf{x})$ is inequality constraints, $h_k(\mathbf{x})$ is equality constraints. Generally, those constraints are nonlinear, so it is difficult to solve them.

The zero-one nonlinear programming problems come from many real-world, such as assignment problems, knapsack problems, portfolio, capital budget, traveling salesman problems [1 − 3] and so on. At present, the zero-one nonlinear

* The work is supported by the National Natural Science Foundation of China under Grant No.60962006 and the Higher School Research Projects of Ningxia under Grant No.2009JY008.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 101–110, 2011.
© Springer-Verlag Berlin Heidelberg 2011

programming problems are solved by the two methods, one is deterministic methods, such as branch and bound algorithm, cutting plane algorithm, filled function method, steepest descent method [2, 4] and so on. The other is the stochastic methods, such as genetic algorithm, simulated annealing algorithm, ant colony algorithm, differential evolution algorithm[5 − 7], etc. This kind of methods can solve the complicated nonlinear optimization problems, but its convergence is very difficult to prove.

We transform zero-one nonlinear programming problems into unconstrained zero-one integer optimization problems by penalty function method [8], i.e.

$$\min F(\mathbf{x}, \delta_t) = f(\mathbf{x}) + \delta_t p(\mathbf{x}), \tag{2}$$

where

$$p(\mathbf{x}) = \sum_{i=1}^{m} (max(0, g_i(\mathbf{x})))^p + \sum_{j=1}^{l} |h_j(\mathbf{x})|^p \tag{3}$$

Clearly, $p(\mathbf{x}) \geq 0$ is the sum of constraint violations, δ_t is penalty factor, p is a positive integer, in this paper given $p = 2$ and $\delta_t \to \infty$. Then we give a PSO algorithm with chaos and gene density mutation to solve the zero-one nonlinear program problems(PSO-CSGDM). In this algorithm, we use chaos to initialize populations and the 0-1 integer operation in updating the positions to produce 0-1 integer particles as well as the fitness variance and gene density strategy to determine whether the population premature phenomenon or not. Numerical simulations show that the proposed algorithm is feasible, effective and is a high precision global optimization algorithm.

2 PSO Algorithm with Chaos and Gene Density Mutation

2.1 Basic PSO

Particle swarm optimization (PSO) was proposed by Eberhart and Kennedy in 1995, it is a kind of swarm intelligence-based computational method [9 − 10], which comes from the study of birds foraging behavior. Suppose that the search space is D-dimension, and then the particle i^{th} of swarm can be represented by a D-dimensional vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$. The velocity of this particle can be represented by another D-dimensional vector $\mathbf{v}_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. The fitness of each particle can be evaluated according to the objective function of optimization problem. The best previously visited position of the particle i^{th} is noted as its individual best position $\mathbf{p}_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$. The position of best individual of the whole swarm is noted as the global best position $\mathbf{p}_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$. At each step, the velocity of particle and its new position will be assigned according to the following two equations:

$$v_{id}(t+1) = wv_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gd}(t) - x_{id}(t)), \tag{4}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1). \tag{5}$$

where $1 \leq d \leq D$, $1 \leq i \leq N$, the superscript t denotes the t^{th} iteration; c_1 and c_2 are positive constants , called the cognitive and social parameter respectively, r_1 and r_2 are random numbers uniformly distributed in the range $(0, 1)$; let the upper limit of the rate of the velocity be v_{max} , when $|v_{id}| > v_{max}$, then $|v_{id}| = v_{max}$, and ω is called inertia weight which has the ability of balancing global search and local search. The experiments indicate that ω will impact on global search ability and local search ability, when ω is higher , the global search ability is strong , but the local search ability is low; whereas the local search ability is strong , but global search ability is low . This paper gives ω from 0.95 linearly decreasing to 0.4 .

2.2 The Equation on the Position of the Particles

a) Initializing population by chaos technique
Firstly, we generate a D-dimensional vector $z_1 = (z_{11}, z_{12}, \cdots, z_{1D})$ and the value of each component is between 0 and 1 , then according to the typical chaotic systems [11] Logistic equation $z_{i+1j} = \mu z_{ij}(1 - z_{ij})$, $j = 1, 2, \cdots, D$, $i = 1, 2, \cdots, N$, where μ is the control parameter, then we get N vectors $z_1, z_2, \cdots, z_N$, then we convert each component of the chaotic variables z_i to decision variables and integer ,i.e.:

$$x_{ij} = x_j^l + [z_{ij} * (x_j^u - x_j^l)]. \tag{6}$$

where $\mathbf{x}_l$ and $\mathbf{x}_u$ are respectively corresponding decision variables of the lower and upper bound, $[\star]$ is rounding integer method.

b)Updating the positions of the particles
In this paper, the position of PSO update formula with rounding integer method, i.e.

$$x_{id}(t+1) = [x_{id}(t) + v_{id}(t+1)], \tag{7}$$

where $x_{id}(t+1)$ is the updated position, $[\star]$ is rounding integer method.

2.3 The Strategy of Overcoming Premature Phenomena

In order to predict the premature convergence of the algorithm, we will use the following two concepts: population fitness variance and gene density.

Definition 1. *Let the particle's number be N, f_i is the fitness of i^{th} particle's, f_{ave} is current average fitness of swarm, the swarm fitness variance δ^2 [12]is defined as:*

$$\delta^2 = \frac{1}{N} \sum_{i=1}^{N} (\frac{f_i - f_{ave}}{f})^2. \tag{8}$$

where $f_{ave} = \frac{1}{N} \sum_{i=1}^{N} f_i$, f is a factor of returning, it can limit δ^2 , and the value of f is defined as:$f = max\{1, max|f_{ave} - f_i|\}, i = 1, 2, \cdots, N$.

Form the definition, the swarm fitness δ^2 shows that all particles convergence degree. The smaller δ^2, the swarm diversity is small, the nearer the particles will rapidly cluster together so that particle swarm cannot be further improved. Therefore,we consider that it has been in premature convergence when $\delta^2 < C(C$ is a given constant).

Definition 2. *(Gene Density)In* $\{0,1\}^D$*,let*

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{pmatrix}$$

where N is the size of this population, D is the dimension of search space,we take each column of X as gene fragment, ρ_j *is the gene density of the* j^{th} *gene fragment in X ,then gene density* ρ_j *is defined as:*

$$\rho_j = \frac{1}{N} \sum_{i=1}^{N} x(i,j). \tag{9}$$

Form the definition, the gene density ρ_j shows that all particles convergence degree. The smaller or larger ρ_j, the genes of the gene fragment in X are very similar, so that the genetic fragment genes lose diversity. At this time, either most of the particles having the same fitness value or most of the gathering in a few specific locations particles of the particles would be premature, namely, the swarm fitness variance is tend to 0 or gene density is approach to 0 or 1. Thereforewe adopt gene density mutation or the strategy of the population re-initialization in order to increase the diversity of population and pseudo-code in table 1. In $\delta^2 < C$ premise,

(i) If $\rho_j > \beta$,it shows that the genes of j^{th} gene fragment are most to 1, we transform many 1 to 0 at a certain probability r, and make sure $\alpha<\rho_j<\beta$. Mutation probability r is ranged from $1-\beta\rho_j$to$1-\alpha\rho_j$ by calculation. Namely, if $x_{ij} = 1$ and $1-\beta\rho_j<rand<1-\alpha\rho_j$,then $x_{ij} = 0$, where $rand$ is a random in $(0,1)$.

(ii) If $\rho_j<\alpha$, it shows that the genes of j^{th}gene fragment are most to 0, we transform many 0 to1 at a certain probability r, and make sure $\alpha<\rho_j<\beta$. Mutation probability r is ranged from $1-\frac{\beta}{1-\rho_j}$ to $1-\frac{\alpha}{1-\rho_j}$ by calculation. Namely, if $x_{ij} = 1$ and $1-\frac{\beta}{1-\rho_j}<rand<1-\frac{\alpha}{1-\rho_j}$,then $x_{ij} = 0$, where $rand$ is a random in $(0,1)$.

(iii) Then reset re-initialization population by chaos technique.

2.4 The Description of the New PSO Algorithm

Now, we describe the proposed algorithm for solving unconstraint problems (2), this algorithm is denoted as PSO-CSGDM.

Table 1. Pseudo-code of overcoming premature phenomena strategy

```
if δ² < C
    for j=1:D
        if ρ_j > β
            for i=1:N
                if x(i,j) = 1 and 1 − βρ_j<r<1 − αρ_j
                    x(i,j)=0;
                end
            end
        end
        if ρ_j < α
            for i=1:N
                if x(i,j) = 0 and 1 − β/(1−ρ_j)<r<1 − α/(1−ρ_j)
                    x(i,j)=1;
                end
            end
        end
    end
else
    Reinitialization population
end
```

step 1. (Initialization)The size of population N the search space dimension D , acceleration factor c_1, c_2 , the maximum iteration time T_{max} ,the threshold C, α, β, penalty factor δ_1 , set $t = 1$.

step 2. According to section 3.2.1, we get the initial population.

step 3. Evaluating the fitness f_i of the particle i^{th}, $i = 1, 2, \cdots, N$, set the initial position of each particle $\mathbf{p}_i$,i.e. $\mathbf{p}_i = \mathbf{x}_i$, the global optimal position $\mathbf{p}_g = argmin(f)$.

step 4. Perform operation of all the particles as follows:

step 4.1. Update the velocity and position according to Eqs.(4)and (7), and put them in a certain extent;

step 4.2. Evaluating the fitness f_i of the particle i^{th} , $i = 1, 2, \cdots, N$, according to Eqs.(2).

step 5. Update the individual optimal position $\mathbf{p}$ and the global optimal position $\mathbf{p}_g$;

step 6. Evaluating the swarm fitness variance by Eqs. (8) and gene density according to Eqs. (9). The particles in premature convergence are mutated by section 3.2.. Then update the individual optimal position $\mathbf{p}$ and the global optimal position p_g ;

step 7. If a stopping criterion is met, then output $\mathbf{p}_g$ and its value; Otherwise let $t = t + 1$, go back to step 4.

3 Numerical Experiments

3.1 Test Function

Example 1[4]: Let us consider the following zero-one programming problem:

$$\begin{cases} \min\ 20x_1 - 10x_2 + 6x_3, \\ s.t.\ \ x_1 + 2x_2 - x_3 \geq 2, \\ \qquad 2x_1 + x_2 + x_3 \leq 6, \\ \qquad x_1, x_2, x_3 \in \{0, 1\}. \end{cases}$$

Example 2[4]: Let us consider the following zero-one programming problem:

$$\begin{cases} \min\ -2x_1 + x_2 - 5x_3 + 3x_4 - 4x_5, \\ s.t.\ \ 3x_1 - 2x_2 + 7x_3 - 5x_4 + 4x_5 \leq 6, \\ \qquad x_1 - x_2 + 2x_3 - 4x_4 + 2x_5 \leq 0, \\ \qquad x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}. \end{cases}$$

Example 3[4]: Let us consider the following zero-one programming problem:

$$\begin{cases} \min\ -6x_1 + 5x_2 + x_1x_3, \\ s.t.\ \ x_2 = x_1^2, \\ \qquad -x_1 + x_2 - 2x_3 \leq 5, \\ \qquad 3x_1 + 5x_2 - 7x_3 \leq -5, \\ \qquad x_1, x_2, x_3 \in \{0, 1\}. \end{cases}$$

Example 4: Let us consider the following zero-one quadratic programming problem, its global optimal value is -27.

$$\min \frac{1}{2} x^T Q x, x \in \{0, 1\}^5.$$

Where

$$Q = \begin{pmatrix} 15 & -4 & 1 & 0 & 2 \\ -4 & -17 & 2 & 1 & 1 \\ 1 & 2 & -25 & -8 & 1 \\ 0 & 1 & -8 & 30 & -5 \\ 2 & 1 & 1 & -5 & -20 \end{pmatrix}$$

Example 5[2]: Let us consider the following nonlinear class of zero-one knapsack problem:

$$\begin{cases} \min \prod_{i=1}^{m} (\sum_{j=1}^{N_i} c_j x_j + d_i), \\ s.t.\ \ \sum_{j}^{N} a_j x_j \geq b, \\ \qquad x_j \in \{0, 1\}, a_j, c_j, b, d_i \geq 0, j = 1, 2, \cdots, N, i = 1, 2, \cdots, m, \\ \qquad b \leq \sum_{j=1}^{N} a_j, \\ \qquad \bigcup_{i=1}^{m} N_i = N. \end{cases}$$

Example 6: Let us consider the following zero-one programming problem:

$$\begin{cases} \min\ f(x) = \frac{1}{2} x^T Q x + d^T x, \\ s.t.\ \ \frac{1}{2} \sum_{j=1}^{n} a_{mj} x_j^2 + d_m^T x \leq t_m, m = 1, 2, \cdots, M, \\ \qquad x_i \in \{0, 1\}, i = 1, 2, \cdots, n. \end{cases}$$

3.2 Parameter Settings

The procedure of the proposed algorithm is compiled with Matlab 7.0. In the procedure, we let the maximum iteration time $T_{max} = 50$, acceleration factor $c_1 = c_2 = 1.8$, the threshold $C = 0.382$, $\alpha = 0.25$, $\beta = 0.75$, the inertia weigh ω from 0.95 linearly decreasing to 0.4.

3.3 Experimental Results and Analysis

In this paper, test process is divided into two steps as following:

(1) The feasibility of PSO-CSGDM
In this section, in order to explain the feasibility of PSO-CSGDM, we will give some computational results of the proposed algorithm through four examples from Reference[4] , each experiment runs 30 times independently, the experimental results were obtained by using PSO-CSGDM algorithm with the above experimental settings. The optimal solution, optimal value, success ratio and CPU for each problem for 30 independently running are summarized in Table 2.

Table 2. Comparison of new algorithm (PSO-CSGDM) and reference [4]

Examples		Optimal solution	Optimal value	Success ratio (%)	CPU(s)
Example 1	PSO-CSGDM	(0, 1, 0)	-10	100	0.0469
	Ref[4]	(0, 1, 0)	-10	-	-
Example 2	PSO-CSGDM	(0, 0, 1, 1, 1)	-6	100	0.0781
	Ref[4]	(1, 1, 0, 1, 1)	-2	-	-
Example 3	PSO-CSGDM	(0, 0, 1)	0	100	0.0156
	Ref[4]	(0, 0, 1)	0	-	-
Example 4	PSO-CSGDM	(0, 1, 1, 0, 1)	-27	100	0.0625

Form Table 2, it can be seen that the new algorithm quickly search the optimal solution for the low-dimension (3˜5 dimension), at the same time, there is no need initial point of the problem. In particular, comparing with reference [4], the optimal solution of the example 2 is $x^* = (1, 1, 0, 1, 1)$, optimal $f(x^*) = -2$, but using the new algorithm can get the optimal solution every experiment, optimal value is $f(x^*) = -6$. It shows that the new algorithm is effective.

(2) The validity of PSO-CSGDM
For this example, we generate some random problems to test the proposed algorithm. All elements of $a(j)_{1\times N}$ and $c(j)_{1\times N}$ are randomly generated in regions $[1, 50]$ and $[1, 20]$, $b = \alpha \sum_{j=1}^{N} a_j$,where α is randomly generated in regions $(0, 1)$, $d_i = \sum_{j\in N_i}(20 - c_j)$, $\bigcup_{i=1}^{m} N_i = N$, $i = 1, 2, \cdots, m$. We transform example 5 into the following formula in order to prevent the solution overflowing.

$$\begin{cases} \min \prod_{i=1}^{m} \frac{\sum_{j=1}^{N_i} c_j x_j + d_i}{\sigma}, \\ s.t.\ \sum_{j}^{N} a_j x_j \geq b, \\ \quad x_j \in \{0,1\}, a_j, c_j, b, d_i \geq 0, j = 1, 2, \cdots, N, i = 1, 2, \cdots, m, \\ \quad b \leq \sum_{j=1}^{N} a_j, \\ \quad \bigcup_{i=1}^{m} N_i = N, \\ \quad \sigma = max_{1 \leq i \leq, m} (\sum_{j=1}^{N_i c_j} + d_i). \end{cases} \tag{10}$$

These problems vary (N, m) in size from $(80, 2)$ to $(10000, 20)$, the experimental results were obtained by using PSO_CSGDM algorithm with the above experimental settings. The minimal optimal value, mean optimal value, standard deviation (Std.) of optimal value and mean CPU for each problem for 30 independently running are summarized in Table 3.

As example 5 belongs to stochastic optimization problem, the initial coefficients are different at each time, but it can be seen from the table 3, under the same conditions, it can be found the optimal solution in less time than reference [2] for different (N, m) by using the new algorithm. Therefore it can be shown that the new algorithm is valid. For this example, we generate some random problems to test the proposed algorithm. These problems vary(N, M)in size from $(10, 5)$ to $(1000, 200)$ .All elements of $(Q_0)_{n\times n}$,$(d_0)_{n\times 1}$ and $(d_m)_{n\times 1}$ are randomly generate in regions $[-1, 0]$,$[-3, -2]$ and $[1, 5]$,respectively, a_{mi} and t_m are generated in $[0, 10]$ and $[400, 500]$, where $j = 1, 2, \cdots, n, m = 1, 2, \cdots, M$ .The experimental results were obtained by using PSO-CSGDM algorithm with the above experimental settings. The minimal optimal value, mean optimal value,

Table 3. The result of the formula (10)

(N,m)	Min	Mean	Std.	CPU(s)
(80, 2)	0.3648	0.4108	0.0277	0.0187
(80, 5)	0.0254	0.0508	0.0155	0.0203
(80, 10)	3.5488e-004	5.9041e-004	2.2784e-004	0.0214
(80, 20)	7.6440e-008	9.0777e-008	7.8526e-008	0.0255
(200, 2)	0.3708	0.4125	0.0173	0.0354
(200, 5)	0.0159	0.0565	0.1032	0.0339
(200, 10)	0.0012	0.0038	0.0019	0.0432
(200, 20)	7.5926e-007	2.7067e-006	2.3232e-006	0.0422
(800, 2)	0.4858	0.5000	0.0080	0.1063
(800, 5)	0.1175	0.1405	0.0110	0.1083
(800, 10)	0.0081	0.0176	0.0035	0.1203
(800, 20)	2.9368e-005	3.1434e-004	1.9001e-004	0.1146
(1000, 10)	0.0164	0.0244	0.0045	0.1401
(1000, 20)	5.7645e-005	2.1210e-004	1.0975e-004	0.1427
(5000, 10)	0.0262	0.0321	0.0033	0.9505
(5000, 20)	3.7183e-004	7.9916e-004	2.1539e-004	0.9411
(10000,10)	0.0297	0.0358	0.0021	2.1432
(10000,20)	7.1415e-004	0.0010	1.8410e-004	2.1182

Table 4. The result of the example 6

(N,M)	Min	Mean	Std.	CPU(s)
(10, 5)	-50.0167	-42.3417	4.1484	0.0073
(20, 10)	-117.4873	-96.5805	7.3489	0.0531
(50, 25)	-161.1217	-119.4951	18.3410	0.2516
(100, 30)	-598.7031	-500.0089	59.1474	0.3896
(120, 40)	-820.2338	-720.8680	75.6152	0.5578
(200, 50)	-2.4063e+003	-2.0462e+003	129.5430	1.2401
(400, 60)	-9.3295e+003	-8.4206e+003	485.8764	13.4203
(600, 100)	-2.0892e+004	-1.9447e+004	838.4629	46.1719
(800, 80)	-3.7420e+004	-3.4856e+004	1.4292e+003	63.2005
(1000, 200)	-5.9212e+004	-5.6006e+004	1.8257e+003	249.0297

standard deviation (Std.) of optimal value and mean CPU for each problem for 30 independently running are summarized in Table 4.

As can be seen from Table 4,our algorithm on randomly generated the same nonlinear programming problem get the optimal solution with increasing scale of problem and the more constrained conditions in a very short time. It shows that the new algorithm is valid.

4 Conclusion

In this paper, we transform zero-one nonlinear programming problems into unconstrained zero-one integer optimization problems by using penalty function. A particle swarm optimization algorithm with chaos and gene density mutation is given to solve unconstrained zero-one nonlinear program problems. We use chaos to initialize populations and use the 0-1 integer operation in updating positions to produce 0-1 integer points. We use the fitness variance and gene density strategy to determine whether the population premature phenomenon or not. If it appears that we use the gene density mutation to increase the population diversity or reset population. Numerical simulations show that PSO-CSGDM algorithm for most test functions is feasible, effective and has high precision.

References

1. Hilier, F.S., Liberman, G.J.: Introduction to mathematical programming. Graw Hill (1981)
2. Kuno, T.: Solving a class of multiplicative programs with 0-1 knapsackconstraints. Journal of Optimization Theory and Applications 103(1), 121–135 (1999)
3. Ying, S., Yuelin, G.: A double objective decision-making model of loans portfolio optimization and particle Swarm optimization algorithm. In: International Conference on Computational Intelligence and Software Engineering, vol. 12 (2009)
4. Anjidani, M., Effati, S.: Steepest descent method for solving zero-one nonlinear programming problems. Applied Mathematics and Computation 193, 197–202 (2007)

5. Sui, Y., Jia, Z., Du, J.: A continuous approach to 0-1 nonlinear problem and its solution with genetic algorithm. Journal of beijing university of technology 34(8), 785–791 (2008) (in china)
6. Chen, G., Liao, X.: A penalty function algorithm for solving 0-1 nonlinear mixed integer programming. Communication on applied mathematics and computation 21(1), 111–115 (2007) (in china)
7. Liu, J., Gao, Y., Li, H.: Improved different evolution algorithm of 0-1 nonlinear programming problems. Computer Engineering and Applications 46(15), 43–46 (2010) (in china)
8. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear programming theory and algorithm. Academic press, New York (1979)
9. Kennedy, I., Eberhart, R.: Particle swarm optimization. In: IEEE Intentional Conference on Neural Networks, pp. 1942–1948 (1995)
10. Shi, Y., Eberhart, R.: A modified swarm optimizer. In: IEEE International Conference of Evolutionary Computation, Anchorage, AK, USA, pp. 125–129 (1998)
11. Caponetto, R., Fortuna, L., Fazzino, S.: Sequences to improve the performance of evolutionary algorithms. IEEE Trans on Evolutionary Computation 7(3), 289–304 (2003)
12. Lv, Z., Hou, Z.: Particle swarm optimization with adaptive mutation. Acta Electronica Sinica 32(3), 416–420 (2004)

A New Binary PSO with Velocity Control

Laura Lanzarini, Javier López,
Juan Andrés Maulini, and Armando De Giusti

III-LIDI (Institute of Research in Computer Science LIDI)
Faculty of Computer Science, National University of La Plata
La Plata, Buenos Aires, Argentina

Abstract. Particle Swarm Optimization (PSO) is a metaheuristic that is highly used to solve mono- and multi-objective optimization problems. Two well-differentiated PSO versions have been defined – one that operates in a continuous solution space and one for binary spaces. In this paper, a new version of the Binary PSO algorithm is presented. This version improves its operation by a suitable positioning of the velocity vector. To achieve this, a new modified version of the continuous gBest PSO algorithm is used. The method proposed has been compared with two alternative methods to solve four known test functions. The results obtained have been satisfactory.

Keywords: Swarm Intelligence, PSO, Binary PSO, Velocity Control.

1 Introduction

Particle Swarm Optimization (PSO) is a metaheuristic proposed by Kennedy and Eberhart [1]. Its operation is based on the simulation of simple social models, and it has been successfully used in function optimization, as well as in neural network training [2].

There are different versions that were developed from the original idea, most of them related to the variation of various algorithm parameters or to the combination and variation of various topologies, sizes, and number of populations [3] [4] [5] [6]. Variations of the algorithm with changes in the way particle velocity is updated have also been proposed, aimed at achieving diversity and avoiding stagnation in the search process [7] [8] [9].

In [10], Kennedy and Eberhart introduced a discrete binary version of PSO for discrete optimization problems. With binary PSO, each particle is represented as a string of zeros and ones. Unlike the continuous version, the discrete version uses the velocity vector as a probability function that allows deciding the value that should take each binary digit that determines the position of the individual.

This paper is organized as follows: In Section 2, the basic components of the PSO algorithm, both in its continuous and binary versions, are described. In Section 3, the method proposed is described, as well as the differences with its original version. In Section 4, the results obtained when comparing the performance of the new algorithm with other binary versions are presented. Finally, Section 5 includes a summary of this paper highlighting the most relevant aspects.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 111–119, 2011.
© Springer-Verlag Berlin Heidelberg 2011

2 Particle Swarm Optimization

2.1 Continuous Particle Swarm Optimization

In PSO, each individual represents a possible solution to the problem and adapts following three factors: its knowledge of the environment (its fitness value), its previous experiences (its memory), and the previous experiences of the individuals in its neighborhood [1]. In this type of technique, each individual is in continuous movement within the search space and never dies.

Each particle is composed by three vectors and two fitness values:

- Vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ stores the current position of the particle
- Vector $pBest_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ stores the best solution found for the particle
- Velocity vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ stores the gradient (direction) based on which the particle will move.
- The fitness value $fitness_x_i$ stores the suitability value of the current solution.
- The fitness value $fitness_pBest_i$ stores the suitability value of the best local solution found so far (vector $pBest_i$)

The position of a particle is updated as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{1}$$

As explained above, the velocity vector is modified taking into account its experience and environment. The expression is:

$$v_i(t+1) = w.v_i(t) + \varphi_1.rand_1.(p_i – x_i(t)) + \varphi_2.rand_2.(g_i – x_i(t)) \tag{2}$$

where w represents the inertia factor [12], φ_1 and φ_2 are acceleration constants, $rand_1$ and $rand_2$ are random values belonging to the (0,1) interval, and g_i represents the position of the particle with the best $pBest$ fitness in the environment of x_i ($lBest$ or $localbest$) or the entire swarm ($gBest$ or $globalbest$). The values of w, φ_1 and φ_2 are important to ensure the convergence of the algorithm. For detailed information regarding the selection of these values, please see [13] and [14].

2.2 Binary Particle Swarm Optimization

PSO was originally developed for a space of continuous values and it therefore poses several problems for spaces of discrete values where the variable domain is finite. Kennedy and Eberhart [10] presented a discrete binary version of PSO for these discrete optimization problems.

In binary PSO, each particle uses binary values to represent its current position and the position of the best solution found. The velocity vector is updated as in the continuous version, but determining the probability that each bit of the position vector becomes 1. Since this is a probability, the velocity vector should

be mapped in such a way that it only contains values within the [0,1] range. To this end, the sigmoid function indicated in (3) is applied to each of its values.

$$v'_{ij}(t) = sig(v_{ij}(t)) = \frac{1}{1 + e^{-v_{ij}(t)}} \tag{3}$$

Then, the particle position vector is updated as follows

$$x_{ij}(t+1) = \begin{cases} 1 \; if \; rand_{ij} < sig(v_{ij}(t+1)) \\ 0 \; if \; not \end{cases} \tag{4}$$

where $rand_{ij}$ is a number ramdomly generated by an uniform pdf in [0,1].

It should be mentioned that the incorporation of the sigmoid function radically changes the way in which the velocity vector is used to update the position of the particle. In continuous PSO, the velocity vector takes on higher values first to facilitate the exploration of the solution space, and then reduces them to allow the particle to stabilize. In binary PSO, the opposite procedure is applied. Each particle increases its exploratory ability as the velocity vector reduces its value; that is, when v_{ij} tends to zero, $\lim_{t\to\infty} sig(v_{ij}(t)) = 0.5$, thus allowing each binary digit to take a value of 1 with a probability of 0.5. This means that it could take on either value. On the contrary, when the velocity vector value increases, $\lim_{t\to\infty} sig(v_{ij}(t)) = 1$, and therefore all bits will change to 1, whereas when the velocity vector value decreases, taking negative values, $\lim_{t\to\infty} sig(v_{ij}(t)) = 0$ and all bits will change to 0. It should be noted that, by limiting the velocity vector values between -3 and 3, $sig(v_{ij}) \in [0.0474, 0.9526]$, whereas for values above 5, $sig(v_{ij}) \simeq 1$ and for values below -5, $sig(v_{ij}) \simeq 0$.

3 Binary PSO with Velocity Control. Method Proposed.

Based on the observations of the behavior of the velocity vector in the binary PSO algorithm defined in [10], and on the importance of correctly calculating the probabilities that allow changing each binary digit, a modified version of the original PSO algorithm to modify the velocity vector is proposed.

Under this new scheme, each particle will have two velocity vectors, $v1$ and $v2$. The first one is updated according to (5).

$$v1_i(t+1) = w.v1_i(t) + \varphi_1.rand_1.(2 * p_i - 1) + \varphi_2.rand_2.(2 * g_i - 1) \tag{5}$$

where the variables $rand_1$, $rand_2$, φ_1 and φ_2 operate in the same way as in (3). The values p_i and g_i correspond to the i^{th} binary digit of the $pBest_i$ and $gBest$ vectors, respectively.

The most significant difference between (3) and (5) is that in the latter, the shift of vector $v1$ in the directions corresponding to the best solution found by the particle and the best global solution does not depend on the current position of the particle. Then, each element of the velocity vector $v1$ is controlled by applying (6)

$$v1_{ij}(t) = \begin{cases} \delta 1_j & if \; v1_{ij}(t) > \delta 1_j \\ -\delta 1_j & if \; v1_{ij}(t) \leq -\delta 1_j \\ v1_{ij}(t) \; if \; not \end{cases} \tag{6}$$

where

$$\delta 1_j = \frac{limit1_{upper_j} - limit1_{lower_j}}{2} \tag{7}$$

That is, velocity vector $v1$ is calculated with (5) and controlled with (6). Its value is used to update velocity vector $v2$, as shown in (8).

$$v2(t+1) = v2(t) + v1(t+1) \tag{8}$$

Vector $v2$ is also controlled as vector $v1$ by changing $limit1_{upper_j}$ and $limit1_{lower_j}$ by $limit2_{upper_j}$ and $limit2_{lower_j}$, respectively. This will yield $\delta 2_j$, which will be used as in (6) to limit the values of $v2$. Then, the new position of the particle is calculated with (4) using the values of $v2$ as arguments of the sigmoid function.

4 Results Obtained

In this section, the performance of the proposed binary PSO variation will be compared to the performance of the method proposed by Kennedy and Eberhart in [10] and the binary PSO defined in [11]. A known set of N-dimensional test functions was minimized.

Forty independent runs were performed for each of the methods using 2,000 iterations. N=3, 5,10 and 20 variables were used. Population size was always 20 particles. The values of $limit1$ and $limit2$ are the same for all variables; $[0; 1]$ and $[0; 6]$, respectively. Thus, probabilities within the interval $[0.0474, 0.9526]$ can be obtained. The values for φ_1 and φ_2 were set at 0.25 in all cases. As regards [10] and [11] methods, velicty limits were set within $[-3, 3]$ in order to keep the same range of probabilities.

The test functions used were Sphere, Rosenbrock, Griewangk, and Rastrigin, which were assigned numbers 1 to 4, respectively.

In table 1, the fitness value for the best solution found by each method is shown, as well as the average best fitness value for all 40 runs.

As it can be observed, the method proposed finds the best solutions and has the lowest average fitness values.

Table 2 indicates whether the results obtained are significant. The symbol ▲ was used to represent that $p - value < 0.05$, indicating that the null hypothesis must be rejected. The test carried out is of the lower tail type, whose null hypothesis states that the mean of the method indicated on the corresponding row is not lower than the mean of the method indicated on the corresponding column. The symbol ▽ indicates that the null hypothesis is not rejected.

Figure 1 shows the plot box diagrams calculated with the best results obtained in each of the 40 runs. Each column corresponds to a different function. Diagrams on the same row correspond to the same number of variables. From top to bottom, rows 1, 2, 3 and 4 correspond to the results obtained when assessing the functions with 3, 5, 10 and 20 variables, respectively. In each figure, methods are numbered from 1 to 3 and correspond to: the method proposed, Binary PSO from [10], and Binary PSO from [11], respectively. As it can be seen, the method proposed offers better solutions than the other two PSO alternatives.

Table 1. Results obtained

		Proposed method		Binary PSO [10]		Binary PSO [11]	
Nro. Var	Nro. Funct	Best Fitness	Average best Fitness	Best Fitness	Average best Fitness	Best Fitness	Average best Fitness
3	1	0	**0**	0	1,2e-09	1,8e-08	6,3e-07
3	2	7,0e-04	**2,8**	1,1e-05	3,0	2,0e-03	5,6
3	3	2,1e-09	**3,3e-03**	2,1e-09	6,8e-03	4,2e-06	8,8e-03
3	4	5,4e-08	**5,4e-08**	5,4e-08	**5,4e-08**	8,9e-06	7,6e-04
5	1	0,0	**3,1e-09**	1,4e-07	1,3e-05	1,7e-04	1,2e-03
5	2	2,2	**28,7**	2,1	111,5	7,2	278,3
5	3	2,6e-09	**8,2e-03**	1,6e-03	2,0e-02	1,9e-02	6,6e-02
5	4	9,0e-08	**1,3e-07**	2,3e-04	5,1e-01	5,0e-01	3,7
10	1	8,2e-05	**9,8e-04**	1,3e-02	7,1e-02	9,7e-02	6,2e-01
10	2	7,3	**141,0**	334,0	2812,8	92013,0	613510,0
10	3	1,3e-02	**7,6e-02**	7,7e-02	1,9e-01	5,2e-01	7,3e-01
10	4	0,8	**4,3**	7,5	15,3	13,7	44,0
20	1	3,3e-01	**8,1e-01**	1,7e+00	3,9e+00	1,2e+01	1,9e+01
20	2	1865,9	**10105**	2,8e+05	1,2e+07	1,2e+08	5,0e+08
20	3	1,4e-01	**2,7e-01**	9,3e-01	1,0e+00	1,3e+00	1,4e+00
20	4	27,7	**43,0**	52,7	88,9	169,8	215,2

Table 2. Results of hypothesis tests

nro Var.	Método	Proposed Binary PSO				Binary PSO [10]				Binary PSO [11]			
3	Proposed binary PSO					▽	▽	▽	▽	▲	▽	▲	▽
3	Binary PSO [10]	▽	▽	▽	▽					▲	▽	▽	▽
3	Binary PSO [11]	▽	▽	▽	▽	▽	▽	▽	▽				
5	Proposed binary PSO					▽	▽	▲	▽	▲	▲	▲	▲
5	Binary PSO [10]	▽	▽	▽	▽					▲	▽	▲	▲
5	Binary PSO [11]	▽	▽	▽	▽	▽	▽	▽	▽				
10	Proposed binary PSO					▲	▲	▲	▲	▲	▲	▲	▲
10	Binary PSO [10]	▽	▽	▽	▽					▲	▲	▲	▲
10	Binary PSO [11]	▽	▽	▽	▽	▽	▽	▽	▽				
20	Proposed binary PSO					▲	▲	▲	▲	▲	▲	▲	▲
20	Binary PSO [10]	▽	▽	▽	▽					▲	▲	▲	▲
20	Binary PSO [11]	▽	▽	▽	▽	▽	▽	▽	▽				

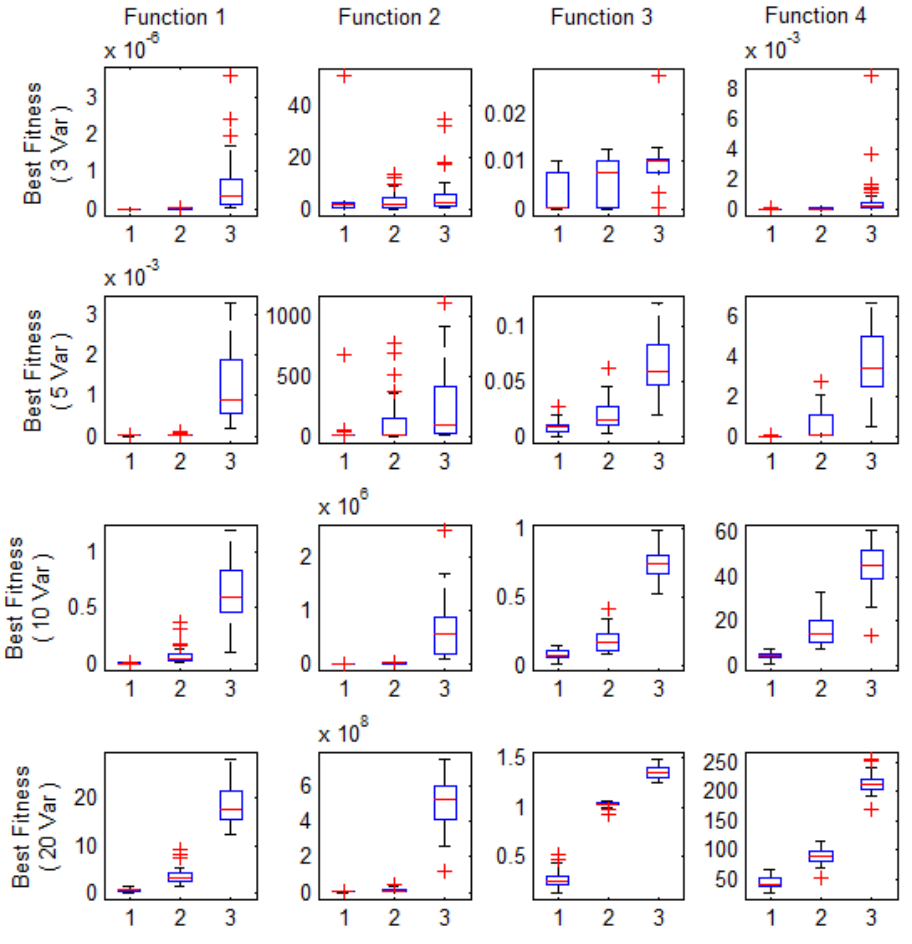


Fig. 1. Boxplots corresponding to the best solutions obtained in each of the 40 independent runs. The method is indicated on the x-axis: 1 = Method proposed, 2 = Binary PSO from [10] and 3 = Binary PSO from [11]. Each row indicates the results obtained with 3, 5, 10 and 20 variables.

Figure 2 is organized in a similar fashion. It shows the boxplots corresponding to the average fitness of each of the 40 runs performed for each function and for each number of variables considered. The populational diversity of each method can be observed there. In general, based on box heights, it can be said that, even though the method from [11] presents the greater inter-quartile ranges, its solutions are the worst. As for the method proposed and the Binary PSO from [10], ranges are equivalent.

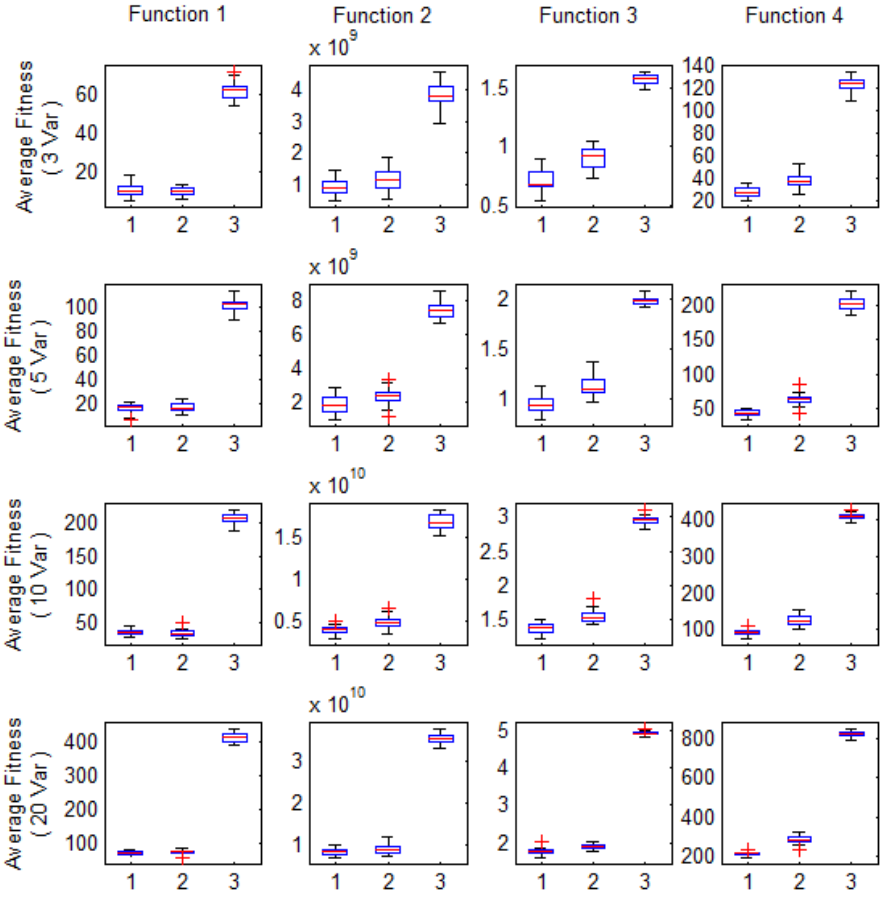


Fig. 2. Boxplots corresponding to the average fitness of each of the 40 independent runs. The method is indicated on the x-axis: 1 = Method proposed, 2 = Binary PSO from [10] and 3 = Binary PSO from [11]. Each row indicates the results obtained with 3, 5, 10 and 20 variables.

5 Conclusions

A variation of the binary PSO method originally proposed in [10] that controls velocity vector changes by using a variation of the continuous PSO method has been presented.

The results obtained by minimizing a set of test functions are better than those obtained with the methods defined in [10] and [11] for all the cases assessed.

Table 2 shows that, as the number of variables used increases, the difference in means becomes more significant.

Figure 1 shows that, in most of the runs, the results obtained with the method proposed have been better than those obtained with the other two methods.

Similarly, Figure 2 shows that the method proposed generates the best population average fitness results, at least for the test functions assessed. If we consider the average fitness inter-quartile range (height of the boxplots in Figure 2), it can be stated that the method proposed in this paper and the Binary PSO from [10] are equivalent, the best solutions being offered by the former.

Currently, our research is focused on measuring the performance of the algorithm proposed with an increasing number of dimensions in the problem, in order to apply the algorithm to real-world problem resolution.

It would also be interesting to analyze its performance using variable-size PSO [6]. This would allow adapting the swarm size based on the complexity of the problem to solve.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, Perth, Australia, vol. IV, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
2. Hu, X., Shi, Y., Eberhart, R.: Recent Advances in Particle Swarm. In: Congress on Evolutionary Computation, CEC 2004, vol. 1, pp. 90–97 (2004)
3. Cagnina, L., Esquivel, S., Coello Coello, C.: A bi-population PSO with a shake-mechanism for solving constrained numerical optimization. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 670–676 (2007)
4. Mohais, A., Mendes, R., Ward, C., Postho, C.: Neighborhood Re-Structuring in Particle Swarm Optimization. In: Zhang, S., Jarvis, R.A. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 776–785. Springer, Heidelberg (2005)
5. Atyabi, A., et al.: Particle Swarm Optimizations: A Critical Review. In: 5th International Conference on Information and Knowledge Technology, Mashad, Iran (2007)
6. Lanzarini, L., Leza, V., De Giusti, A.: Particle Swarm Optimization with Variable Population Size. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 438–449. Springer, Heidelberg (2008)
7. Riget, J., Vesterstrom, J.: A Diversity-Guided Particle Swarm Optimizer – the ARPSO. EVALife Project Group Department of Computer Science (2002)
8. Clerc, M.: Stagnation Analysis in Particle Swarm Optimisation or What Happens When Nothing Happens.Department of Computer Science; University of Essex. Technical Report CSM-460 (2006)
9. Peer, E., Van den Bergh, F., Engelbrecht, A.: Using Neighbourhoods with the Guaranteed Convergence PSO. In: Swarm Intelligence Symposium, SIS 2003, pp. 235–242 (2003)
10. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: Proc. of the World Multiconference on Systemics, Cybernetics and Informatics (WMSCI), pp. 4104–4109 (1997)
11. Khanesar, M., Teshnehlab, M., Shoorehdeli, M.: A novel Binary Particle Swarm Optimization. In: 18th Mediterranean Conference on Control and Automation, Athens, pp. 1–6 (June 2007)

12. Shi, Y., Eberhart, R.: Parameter Selection in Particle Swarm Optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998) ISBN 3-540-64891-7
13. Clerc, M., Kennedy, J.: The particle swarm – explosion, stability and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
14. Van den Bergh, F.: An analysis of particle swarm optimizers. Ph.D. dissertation. Department Computer Science. University Pretoria. South Africa (2002)

Adaptive Particle Swarm Optimization Algorithm for Dynamic Environments

Iman Rezazadeh[1], Mohammad Reza Meybodi[2], and Ahmad Naebi[1]

[1] Department of Computer Engineering, Qazvin Islamic Azad University, Qazvin, Iran
[2] Department of Computer Engineering and Information Technology,
Amirkabir University of Technology, Tehran 15914, Iran
{Eiman.rezazadeh,ahmad.naebi}@gmail.com, mmeybodi@aut.ac.ir

Abstract. Many real world optimization problems are dynamic in which global optimum and local optimum change over time. Particle swarm optimization has performed well to find and track optimum in dynamic environments. In this paper, we propose a new particle swarm optimization algorithm for dynamic environments. The proposed algorithm utilizes FCM to adapt exclusion radios and utilize a local search on best swarm to accelerate progress of algorithm and adjust inertia weight adaptively. To improve the search performance, when the search areas of two swarms are overlapped, the worse swarms will be removed. Moreover, in order to track quickly the changes in the environment, all particles in the swarm convert to quantum particles when a change in the environment is detected. Experimental results on different dynamic environments modeled by moving peaks benchmark show that the proposed algorithm outperforms other PSO algorithms, for all evaluated environments.

Keywords: MPB, Dynamic Environment, PSO, Moving Peaks.

1 Introduction

PSO is relatively a new heuristic search method, this algorithm has successfully been utilized in variety of applications such as pattern recognition, image processing, machine learning, etc. PSO is an optimized algorithm which is inspired from social and group life of animals like birds in order to get the optimum solution. In PSO a group of particles are located in search area. Giving that each particle presents a candidate solution of the optimization problem. Location of each particle is driven from the best location which has ever met (in terms of self-experience) and location of the best neighborhood particles (in terms of neighborhood experience).

The applications in which the evolutionary algorithms are applied are divided in to two parts: Static and Dynamic. Majority of the real world problems have dynamic nature and are subjected to change over the time. For example, the new tasks that are received continuously and must be scheduled. Parameters which effect the dynamic environment, the frequency of the change, severity of the change, predictability of the change, Cycle length and cycle accuracy. Dynamic environments are divided in to four sections based on defined settings: constant (identical change in each cycle), periodical, homogeneous and alternating [6].

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 120–129, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Drawbacks of The PSO in dynamic environments are: old memory and diversity loss which are explained in next paragraphs as bellow:

In the case of any change in environment, the particle's memory is not true anymore and can have a very bad effect on search process. This problem can be solved by two methods: either re-evaluating the memory or forgetting the memory. In re-evaluating, the memory of each particle is verified in each stage. And in forgetting the memory the current location of each particle is replaced with its memory and overall optimization is updated accordingly. Diversity loss occurs when the swarm converges on a few peaks in the landscape and loses its ability to find new peaks, which is required after the environment changes. There are two approaches to deal with diversity losing problem. In the first approach, a diversity maintenance mechanism runs periodically (or when a change is detected) and re-distributes the particles if the diversity falls below a threshold. In the second approach, diversity is always monitored and as soon as it falls below a threshold, the swarm will be re-diversified.

The rest of the paper is organized as follows. In Section 2, related works on dynamic environments are reviewed, in section 3, the proposed algorithm is presented. In Section 4, presents the experimental results of the proposed algorithm along with comparison with alternative approaches from the literature. Finally section 5 includes the conclusion of the present paper.

2 Related Works

MPSO is suggested by Blackwell and Branke [4.5]. The particles in MPSO are divided to M independent groups. Each group contains a fixed number of particles. Information sharing in each group is done in global manner. This mechanism keeps the diversity in two levels: group is divided into sub-groups which penetrate in different sections of search area (diversity between the groups). And each sub-group contains some quantum particles which provide diversity inside the group. In [10], the effectiveness of this algorithm is analyzed and demonstrated that the quantum articles used in this algorithm are only useful when the environment is subject to change and doesn't have much efficiency in other cases. In [11] because of less efficiency of quantum particles, two types of strategies are utilized: in the first strategy, in the time of detecting any change in environment, the particles are divided to three parts: the first part remains without change, the second part like quantum particles in a cloud are assigned by value of centrality of the best particle and radius of "r", and the third part are distributed in whole of the environment. In second strategy useless swarms are detected using fuzzy logic and stopped in order not to waste the system's resources. In [12] a Cauchy Mutation is utilized for detecting the changes in environment, instead of quantum particles. Also the small neighborhoods are used inside the groups. In this work, some of the particles are kept away from center when the group is going to be convergent in order to keep the diversity.

SPSO [5] distributes the particles dynamically between the types. SPSO is extended based on the theory of the types. The limitation of the types depends on parameter r_s which represents the measured radiuses in Euclidean distance from the center of types to its borders. The center of a type so called the "type's core", usually is a particle with the best fitness. All of the particles within the r_s radius from the core are categorized as a similar type.

In FMSO [7], two search algorithm, one in parent group and another in child group are utilized. The global search function is utilized in parent group in order to keep the diversity and finding the probable areas in search environment. Meanwhile hand the child groups are used as local explorer. FMSO starts with a parent group which does the search function in environment In each level if the best founded location in parent's group is improved a child group in centrality of the founded location and radius of r_s from this location is created. Kamosi improved this procedure in [9].

Hashemi and Meybodi introduced cellular PSO, a hybrid model of cellular automata and PSO[3]. In cellular PSO, a cellular automaton partitions the search space into cells. At any time, in some cells of the cellular automaton a group of particles search for a local optimum using their best personal experiences and the best solution found in their neighborhood cells. To prevent losing the diversity, a limit on the number of particles in each cell is imposed.

In KPSO [1], in order to divide the problem into sub-problems consecutive repetition, all particles in problem domain are clustered and each cluster performs the search process independently. At first in this mechanism, the grouping process is performed and stays unchanged for several repetitions. So, there will be enough time for algorithm to do the search. Also because of utilization of the clustering, spatial location of the particles in various groups are considered. Drawback of this approach is defining the suitable number of clusters. In [8], Lee and Yong cluster the particles using fuzzy clustering and grouped the particles using the result of the clustering then according to the progress of the algorithm compound the clusters. If the particles of two clusters are a few, and near to each other, these two clusters are compound. Also in order to solve the problem of two steps forward and two steps back in PSO, when the best particles are updating, the dimensions of that particles are updated one by one. In [13] Lee and Yong, at first all of the particles are distributed in the environment and in each repetition the neighboring particles make one group. This process continues until there is no single particle group in the environment. If two groups are closer than a threshold, then they compound together. And if the number of particles in one group is so many then the worse particles are deleted. If any changes have been made in the environment, again a series of the particles are produced in order to keep the diversity of the particles.

3 Proposed Method

In our proposed model, the inertia weight has been adjusted adaptively. If the particles of swarm have been improved in previous iteration, it shows that the previous movement of the particles is good and they should continue their pervious movement. So the inertia weight must be high.

If the particles of swarm have been failed, it shows that their previous movement isn't good enough and it is better that these particles don't continue the previous movement so the inertia weight must be decreased. But if all swarms utilize only this method, they may fall in local optimum. To prevent this, we must not let algorithm reduce all swarms inertias weight more than enough.

The groups that have better fitness may be closer to global optimum. So these groups have to have low inertial weight to do local search for groups that have worst

fitness may be far from global optimum and therefore may fall in local optimum. In order to avoid from local optimum and find the global optimum, they should have bigger inertial weight to do global search. In this way, the group descending sorted and ordered with number at first to last. First group has the order of one and the last group has the maximum order. The rank of each group is divided to the number of groups and selected as the minimum of group inertial weight. So this way, inertial weight of each group is calculated according to (1), (2):

$$w_{min}^{i} = \frac{rank_i}{swarm_{size}} * w_{max} \tag{1}$$

$$w_i = w_{min}^{i} + \left(w_{max} - w_{min}^{i}\right) * \left(\frac{number\ of\ improved\ particle\ in\ swarm_i}{swarm_{size}}\right) \tag{2}$$

At first the groups are generated randomly, and start the search. Each group is composed with some PSO particles. Since small neighborhoods causes reduction of convergence's speed, and increase in diversity, performs well in complex environments, this algorithm utilize the small neighborhoods. The less the number of particles for the fixed number of groups, the less the number of evaluations which is leading to keep the environment unchanged for more iterations and effective search will be performed in environment. The groups are categorized to two categories: Converged and Free. If the numbers of free groups in the environment are less than a threshold, one group will be added to the existing groups, and if this number be more than a threshold, the worst group deleted from search domain. At each iteration, velocity and position of a particle *i* in each swarm is updated using its best personal position ($pbest_i$) and the best position found by the swarm (*gbest*) according to(5) and (6), respectively. If the fitness of the new position of particle *i* is better than its best personal position ($pbest_i$), $pbest_i$ will be updated to the new position. Likewise, the best position found in the swarm (*gbest*) will be updated.

Since searching an area with more than one swarm is not very useful, at the end of each iteration every two swarms are checked whether they are searching in the same area or not. Two swarms will be searching in the same area or they are colliding, if the Euclidian distance between their attractors is less than a specified threshold r_{excl}. If a collision between two swarms is detected, the swarm whose attractor is worse than the others will be destroyed.

In the proposed algorithm, when an environment change is detected, particles in the swarm re-evaluate their best personal position (*pbest*) and the particles in the swarms change their behaviors in the following iteration after a change is detected in the environment. They will set their new positions to a random location in a hyper sphere with radius r_q centered at their swarm's attractor. Then they will update their best personal positions (*pbest*) and update the swarm's attractor.

In previous works exclusion distance adjusted without considering environment situation. In purposed algorithm this value is adjusted with considering environment conditions and particles density. Current particles are clustered and then mean of minimum distance between each cluster with other clusters are calculated and used as the r_{excel} according to (3) and (4). For clustering we use FCM and number of clusters equal to the swarm size.

$$mindist_i = \min(dist(center_i, center_j)), \text{where } 1<I,j<n,i<>j. \tag{3}$$

$$r_{excel} = \frac{mean(mindist)}{2^{-n}+2}. \tag{4}$$

$$V_i(t+1)=wv_i(t)+c_1r_1(pbest_i-p_i)+c_2r_2(gbest_i-p_i). \tag{5}$$

$$P_i(t+1)=P_i(t)+v_i(t+1). \tag{6}$$

```
Procedure Proposed Algorithm
 begin
  Initialization swarms
  Repeat
   adopt swarms number according to free swarms
   for each swarm do
    if a change is detected in the environment then
     evaluate all pbests then regenerate particles
randomly in a hyper sphere with radius r centered at
gbest and update pbests
      update gbest
    else
     if swarm is the best swarm
      do local search around gbest in hyper sphere with
radius r and update only gbest
     end if
     For each particle in swarm do
      update particle position according to eq.1 and eq.2
and eq.5 and sq.6
      update pbest and gbest
     end-for
    end-if
   test for converge
  end-for
   r_excl=fcmdist
  for each pair of swarms m and n, m<>n do
   test for conflict if there is delete worse swarm
  end-for
  until a maximum number of fitness evaluations is
reached
end.
```

Fig. 1. Pseudocode of the proposed algorithm

```
  Function distfcm
    for all particle do
      data_i=position particle_i
    end.
    center=fcm(data,swarmsize)
    for each center
      dist_i=calculate distance with nearest center
    end
    calculate r_exce using eq2
  end.
```

Fig. 2. Pseudo code of the FCM distance

4 Experimental Studying

In this section, we first describe moving peaks benchmark [2] on which the proposed algorithms is evaluated. Then, experimental settings are described. Finally, experimental results of the proposed algorithm are presented and compared with alternative approaches from the literature.

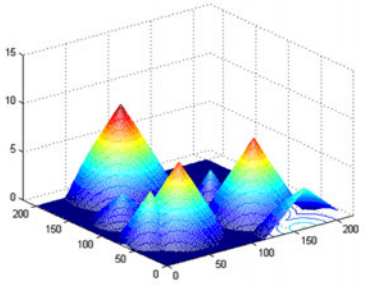


Fig. 3. Moving peaks benchmark

Table 1. Default settings of moving peaks benchmark

Parameter	Value
number of peaks m	10
frequency of change f	every 5000 FEs
height severity	7.0
width severity	1.0
peak shape	cone
shift length s	1.0
number of dimensions D	5
cone height range H	[30.0, 70.0]
cone width range W	[1, 12]
cone standard height I	50.0
search space range A	[0, 100]

4.1 Moving Peaks Benchmark

Moving peaks benchmark (Fig. 3) [2] is widely used in the literature to evaluate the performance of optimization algorithms in dynamic environments [14]. In this benchmark, there are some peaks in a multi-dimensional space, where the height, width, and position of each peak alter when the environment changes. Unless stated otherwise, the parameters of the moving peaks benchmark are set to the values presented in Table 1.

In order to measure the efficiency of the algorithms, offline error that is the average fitness of the best position found by the swarm at every point in time (7), is used [15].

$$offline_{error} = \frac{1}{T}\sum_{t=1}^{T}\left(F\left(global_{opt}(t)\right) - F\left(swarm_{best}(t)\right)\right). \quad (7)$$

where T is the maximum iteration, and $swarm_{best}(t)$ is best position solution by the swarm at iteration t.

4.2 Experimental Settings

For the proposed algorithms the total of acceleration coefficients c_1 and c_2 are set to 1.496180 and the inertial weight w is set to [0, 0.802828]. The number of particles in the swarm is set to 3 particles. The radius of quantum particles (r_q) is set to 0.5. The proposed algorithm is compared with mQSO[5] and FMSO [7], and cellular PSO [3], and kamosi[9]. For mQSO we adapted a configuration 10(5+5^q) which creates 10 swarms with 5 neutral (standard) particles and 5 quantum particles with r_{cloud}=0.5 and r_{excl}= r_{conv} =31.5, as suggested in [5]. For FMSO, there are at most 10 child swarms each has a radius of 25.0. The size of the parent and the child swarms are set to 100 and 10 particles, respectively[7]. For cellular PSO, a 5-Dimensional cellular automaton with 10^5 cells and Moore neighborhood with radius of two cells is embedded into

the search space. The maximum velocity of particles is set to the neighborhood radius of the cellular automaton and the radius for the random local search (r) is set to 0.5 for all experiments. The cell capacity θ is set to 10 particles for every cell. Moreover, all particles perform a local search in the iteration after a change in the environment is detected [3]. For the Kamosi algorithms the acceleration coefficients c_1 and c_2 are set to 1.496180 and the inertial weight w is set to 0.729844. The number of particles in the parent swarm and the child swarms (π) are set to 5 and 10 particles, respectively. The radius of the child swarms (r), the minimum allowed distance between two child swarm (r_{excl}) and the radius of quantum particles (r_q) are set to 30.0, 30.0, and 0.5, respectively[9].

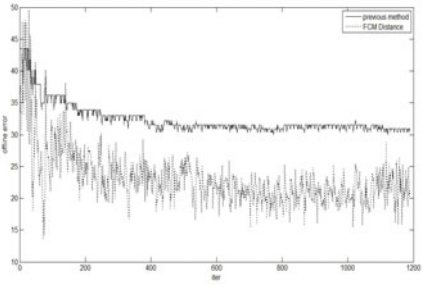


Fig. 4. Comparison on FCM_Distance and previous method in environment with 10 peaks and frequency=500

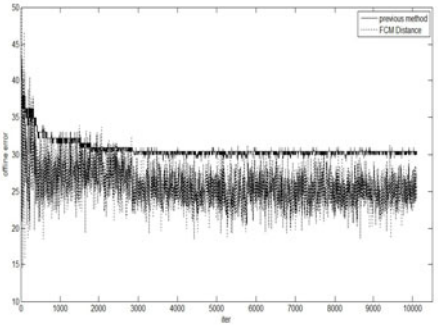


Fig. 5. Comparison on FCM_Distance and previous method in environment with 10 peaks and frequency=5000

4.3 Experimental Results

For all algorithms we reported the average offline error and 95% confidence interval for 100 runs. Offline error of the proposed algorithm, mQSO10($5+5^q$) [5], FMSO [7], cellular PSO[3], and kamosi[9] for different dynamic environment is presented in table 2 to table 6. For each environment, result of the best performing algorithm(s) with 95% confidence is printed in bold. As depicted in the table 2 to table 6, the proposed algorithm outperforms other tested PSO algorithms, including FMSO, for all environments. Moreover, the difference between offline error of the proposed algorithm and the next best algorithm decreases as the environment changes less frequently from f=500 (table 2) to f=10000 (table 6). This is because the proposed algorithm uses less number of particles and so it doesn't waste fitness evaluation and also because the adaptation inertia weight using purposed method , swarm converge very quickly to optimum hence quickly finds better solutions than other algorithms after a change occurs in the environment, especially at the early iterations.

Furthermore, in the proposed algorithm the number of swarms converges to the number of peaks in the environment. This will help the proposed algorithm to track the changes more effectively since there will be a swarm on each peak. For adjusting

exclusion distance adaptively, algorithm can adjust exclusion distance corresponding to the environment and if peaks are near to each other doesn't delete swarm that converged to those peaks.

Table 2. Offline error ±Standard Error for f =500

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**4.81±0.14**	5.46±0.30	33.67±3.42	27.58±0.94	13.46±0.7
5	**4.95±0.11**	5.48±0.19	11.91±0.76	19.45±0.45	9.63±0.49
10	**5.16±0.11**	5.95±0.09	9.62±0.34	18.26±0.32	9.42±0.21
20	**5.81±0.08**	6.45±0.16	9.07±0.25	17.34±0.30	8.84±0.28
30	**6.03±0.07**	6.60±0.14	8.80±0.21	16.39±0.48	8.81±0.24
40	**6.10±0.08**	6.85±0.13	8.55±0.21	15.34±0.45	8.94±0.24
50	**5.95±0.06**	7.04±0.10	8.72±0.20	15.54±0.26	8.62±0.23
100	**6.08±0.06**	7.39±0.13	8.54±0.16	12.87±0.60	8.54±0.21
200	**6.20±0.04**	7.52±0.12	8.19±0.17	11.52±0.61	8.28±0.18

Table 3. Offline error ±Standard Error for f =1000

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**2.72±0.04**	2.90±0.18	18.60±1.63	14.42±0.48	6.77±0.38
5	**2.99±0.09**	3.35±0.18	6.56±0.38	10.59±0.24	5.30±0.32
10	**3.87±0.08**	3.94±0.08	5.71±0.22	10.40±0.17	5.15±0.13
20	**4.13±0.06**	4.33±0.12	5.85±0.15	10.33±0.13	5.23±0.18
30	**4.12±0.04**	4.41±0.11	5.81±0.15	10.06±0.14	5.33±0.16
40	**4.15±0.04**	4.52±0.09	5.70±0.14	9.85±0.11	5.61±0.16
50	**4.11±0.03**	4.57±0.08	5.87±0.13	9.54±0.11	5.55±0.14
100	**4.26±0.04**	4.77±0.08	5.83±0.13	8.77±0.09	5.57±0.12
200	**4.21±0.02**	4.76±0.07	5.54±0.11	8.06±0.07	5.50±0.12

Table 4. Offline error ±Standard Error for f =2500

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**1.06±0.03**	1.10±0.06	7.64±0.64	6.29±0.20	4.15±0.25
5	**1.55±0.05**	1.68±0.16	3.26±0.21	5.03±0.12	2.85±0.24
10	**2.17±0.07**	2.33±0.06	3.12±0.14	5.09±0.09	2.80±0.10
20	**2.51±0.05**	2.79±0.10	3.58±0.13	5.32±0.08	3.41±0.14
30	**2.61±0.02**	2.88±0.09	3.63±0.10	5.22±0.08	3.62±0.12
40	**2.59±0.03**	2.86±0.07	3.55±0.10	5.09±0.06	3.84±0.12
50	**2.66±0.02**	2.97±0.06	3.63±0.10	4.99±0.06	3.86±0.10
100	**2.62±0.02**	3.00±0.05	3.58±0.08	4.60±0.05	4.10±0.11
200	**2.64±0.01**	2.99±0.04	3.30±0.06	4.34±0.04	3.97±0.10

Table 5. Offline error ±Standard Error for f =5000

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**0.53±0.01**	0.56±0.04	3.82±0.35	3.44±0.11	2.55±0.12
5	**1.05±0.06**	1.06±0.06	1.90±0.08	2.94±0.07	1.68±0.11
10	**1.31±0.03**	1.51±0.04	1.91±0.08	3.11±0.06	1.78±0.05
20	**1.69±0.05**	1.89±0.04	2.56±0.10	3.36±0.06	2.61±0.07
30	**1.78±0.02**	2.03±0.06	2.68±0.10	3.28±0.05	2.93±0.08
40	**1.86±0.02**	2.04±0.06	2.65±0.08	3.26±0.04	3.14±0.08
50	**1.95±0.02**	2.08±0.02	2.63±0.08	3.22±0.05	3.26±0.08
100	**1.95±0.01**	2.14±0.02	2.52±0.06	3.06±0.04	3.41±0.07
200	**1.90±0.01**	2.11±0.03	2.36±0.05	2.84±0.03	3.40±0.06

Table 6. Offline error ±Standard Error for f =10000

M	Proposed algorithm	Kamosi	mQSO10(5+5^q)	FMSO	Cellular PSO
1	**0.25±0.006**	0.27±0.02	1.90±0.18	1.90±0.06	1.53±0.12
5	**0.57±0.03**	0.70±0.10	1.03±0.06	1.75±0.06	0.92±0.10
10	**0.82±0.02**	0.97±0.04	1.10±0.07	1.91±0.04	1.19±0.07
20	**1.23±0.02**	1.34±0.08	1.84±0.08	2.16±0.04	2.20±0.10
30	**1.39±0.02**	1.43±0.05	2.00±0.09	2.18±0.04	2.60±0.13
40	**1.37±0.01**	1.47±0.06	1.99±0.07	2.21±0.03	2.73±0.11
50	**1.46±0.01**	1.47±0.04	1.99±0.07	2.60±0.08	2.84±0.12
100	**1.38±0.01**	1.50±0.03	1.85±0.05	2.20±0.03	2.93±0.09
200	**1.36±0.01**	1.48±0.02	1.71±0.04	2.00±0.02	2.88±0.07

5 Conclusion

In this paper, we proposed a new multi-swarm PSO algorithm for dynamic environments. The proposed PSO adjust exclusion distance considering environmental conditions. In order to do this all of the present particles are clustered and then distances between centers of clusters are used to adjust exclusion distance. In order to improve efficiency of algorithm we used a local search around best particle of best swarm and also inertia weight adjusted according to the swarm progress so convergence of algorithm is accelerated. And adjusting exclusion radius using clustering particle causes algorithm don't delete swarms that converged to the peaks when peaks are near of each other. To prevent redundant search in the same area if two swarms collide the one with the worse fitness will be removed. In addition, to track the local optima after detecting a change in the environment, particles in each swarm temporarily change their behavior to quantum particles and perform a random search around the swarm's attractor. Results of the experiments show that for all tested environments the proposed algorithm outperforms all tested PSO algorithms, the previously presented multi-swarm algorithm with the similar approach.

References

1. Passaro, A., Starita, A.: Particle Swarm Optimization for Multimodal Functions: a Clustering Approach. Journal of Artificial Evolution and Applications 2008, article id 482032 (2008)
2. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Congress on Evolutionary Computation CEC 1999, vol. 3, pp. 1875–1882 (1999)
3. Hashemi, A.B., Meybodi, M.R.: Cellular PSO: A PSO for Dynamic Environments. In: Cai, Z., Li, Z., Kang, Z., Liu, Y. (eds.) ISICA 2009. LNCS, vol. 5821, pp. 422–433. Springer, Heidelberg (2009)
4. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments. Springer, Berlin (2007)
5. Blackwell, T., Branke, J., Li, X.: Particle swarms for dynamic optimization problems. In: Swarm Intelligence: Introduction and Applications, Berlin, Germany (2008)
6. Branke, J.: Evolutionary optimization in dynamic environments, http://www.amazon.com/Evolutionary-Optimization-Environments-Algorithms-Computation/dp/0792376315
7. Li, C., Yang, S.: Fast Multi-Swarm Optimization for Dynamic Optimization Problems. In: Fourth International Conference on Natural Computation, Jinan, Shandong, China, vol. 7, pp. 624–628 (2008)
8. Li, C., Yang, S.: A Clustering Particle Swarm Optimizer for Dynamic Optimization. IEEE, Los Alamitos (2009) 978-1-4244-2959-2/09/$25.00_c
9. Kamosi, M., Hashemi, A.B., Meybodi, M.R.: A New Particle Swarm Optimization Algorithm for Dynamic Environments. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) SEMCCO 2010. LNCS, vol. 6466, pp. 129–138. Springer, Heidelberg (2010)
10. del Amo, I.G., Pelta, D.A., González, J.R., Novoa, P.: An Analysis of Particle Properties on a Multi-swarm PSO for Dynamic Optimization Problems. In: Meseguer, P., Mandow, L., Gasca, R.M. (eds.) CAEPIA 2009. LNCS, vol. 5988, pp. 32–41. Springer, Heidelberg (2010) ISBN:3-642-14263-X 978-3-642-14263-5
11. Novoa-Hernández, P., Pelta, D.A., Corona, C.C.: Improvement Strategies for Multi-swarm PSO in Dynamic Environments. In: Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, Granada, Spain, May 12-14 (2010)
12. Hu, C., Wu, X., Wang, Y., Xie, F.: Multi-swarm Particle Swarm Optimizer with Cauchy Mutation for Dynamic Optimization Problems. In: Cai, Z., Li, Z., Kang, Z., Liu, Y. (eds.) ISICA 2009. LNCS, vol. 5821, pp. 443–453. Springer, Heidelberg (2009)
13. Yang, S., Li, C.: A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments. IEEE Transactions on Evolutionary Computation 14(6) (December 2010)
14. Moser, I.: All Currently Known Publications On Approaches Which Solve the Moving Peaks Problem. Swinburne University of Technology, Melbourne (2007)
15. Branke, J., Schmeck, H.: Designing evolutionary algorithms for dynamic optimization problems. In: Advances in Evolutionary Computing: Theory and Applications, pp. 239–262. Springer-Verlag New York, Inc., New York (2003)

An Improved Particle Swarm Optimization with an Adaptive Updating Mechanism

Jie Qi and Yongsheng Ding

College of Information Science and Techenology, Donghua University, Shanghai, China
jieqi@dhu.edu.cn

Abstract. Premature convergence when solving multimodal problems is still the main limitation which affects the performance of the PSO. To avoid of premature, an improved PSO algorithm with an adaptive updating mechanism (IPSO) is proposed in this paper. When the algorithm converges to a local optimum, the updating mechanism begins to work so that the stagnated algorithm obtains energy for optimization. That is, the updating mechanism refreshes the swarm and expands the range for exploration. In this way, the algorithm can achieve a good balance between global exploration and local exploitation by the combination of the basic PSO evolution and updating mechanism. The proposed method is tested with a set of 10 standard optimization benchmark problems and the results are compared with those obtained through other 4 existing PSO algorithms. The simulation results elucidate that the proposed method produces the near global optimal solution, especially for those complex multimodal functions whose solution is difficult to be found by the other 4 algorithms. It is also observed from the comparison the IPSO is capable of producing a quality of optimal solution with faster rate.

Keywords: particle swarm optimization, updating mechanism, benchmark function, swarm intelligence.

1 Introduction

Since particle swarm optimization (PSO) was introduced by Kennedy and Eberhart [1], due to its effectiveness and simplicity in implementation, many successful applications have been seen in solving real world optimization problems [2-4]. As the original PSO may easily get trapped in the local optima when solving complex multimodal problems, many variants have been proposed to improve the performance [5-8]. The main variations in the algorithm are summarized: (1)Parameters adjustment. There are linearly [6] or nonlinearly [9] decreasing inertia weight, time-varying acceleration coefficients (TVAC), the strategy to adjust the parameters with fuzzy methods using feedback information from evolutionary state [8] and the self-adaptive method by encoding the parameters into the particles and optimizing them together with the position during run time [10]. (2) Neighborhood topology. Improving PSO's performance by designing different types of topologies has been an active research direction. Swarm topology can change the algorithm's convergence properties by influencing the information transfer mode among particles [5]. Low connected topologies result in more exploratory

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 130–137, 2011.
© Springer-Verlag Berlin Heidelberg 2011

behavior than highly connected ones. The main static topologies not vary over a run include gbest, lbest, pyramid, star, von Neumann, random topology, and so on [11]. Recent research suggests that time-varying and self-adaptive topologies can be competitive to static ones [5, 7]. (3) Restart mechanism. Through re-initializing particles' velocities or positions or perturbation of the current best particle, restart mechanism is introduced into the algorithm to prevent the algorithm from stagnation. In the self-organized hierarchical PSO with time-varying acceleration coefficients (HPSOTVAC) [6], a dimension of a particle's velocity is reinitialized as it is very close to zero. In [8], elitist learning strategy was used in which one dimension of the globally best particle is choose randomly and is reinitialized within the feasible space. In [12], a restart method is realized by relocating the particles when they are too close to each other. (4) Hybrid particle swarm. Some researchers investigated hybridization by combining PSO with other search techniques [5].

In this paper, we propose an improved PSO algorithm embedded an adaptive updating mechanism (IPSO) which can be included in the variation (3): Restart mechanism. However, the key to obtain an effective algorithm is to determine when to start the mechanism and how to develop a good mechanism. We present a new view of system and energy to consider the algorithm. The original PSO can be seen as a system with kinetic and potential energy. After initialization, the energy is decreasing as the algorithm converging. From the view of system and energy, the energy has been consumed since it has been converted to the search campaign in the solution space. If the algorithm converges prematurely to a local optimum, the energy trends to zero so that the algorithm stagnates. To address this case, an update mechanism is introduced to the algorithm. When the algorithm converges to a certain extent, i.e., the energy is below a critical value, the updating mechanism is triggered and new energy is injected into swarm so that the algorithm regain momentum and PSO evolution runs again. The IPSO is easy to implement and only an embedded updating mechanism is required. We testify the performance of the proposed algorithm on a 10 benchmark functions and provide comparisons with 4 classical PSO variants. The simulation results demonstrate that the IPSO has comparable or better performance of global optimization to avoid the local optima compared with other PSO variants experimented in this paper.

The paper is organized as follows. In section 2, the basic PSO is briefly introduced and the IPSO is described in detail. Section 3 gives experimental results of the IPSO and other 4 PSO variants to solve 10 benchmark functions. Finally, conclusions are drawn in section 4.

2 The Improved Particle Swarm Optimization Embedded an Adaptive Updating Mechanism

PSO is a population based optimization technique, where a swarm of particles $I = \{1,...,i,...,N\}$ is randomly initialized and the algorithm searches for optima by updating generations. In PSO, each particle *i* is associated with four vectors: position vector $\{x_{id}\}$ representing a potential solution; velocity vector $\{v_{id}\}$ measuring the direction and distance of one step move; historical best position vector (pbests) $\{pb_{id}\}$ recording the best position particle *i* ever visited; and the neighbor best position vector

(gbests) $\{pg_{id}\}$ denoting the best position in the neighborhood, where d=1,…,D is the dimensions of the solution space.

During the iteration process, the velocity and position of particle i on dimension d are updated as [5]

$$v_{id} = v_{id} + c_1 rand_1(pb_{id} - x_{id}) + c_2 rand_2(pg_{id} - x_{id}) . \quad (1)$$

$$x_{id} = x_{id} + v_{id} . \quad (2)$$

where c_1 and c_2 are the acceleration coefficients, and $rand_1$ and $rand_2$ are two uniformly distributed random numbers independently generated within [0, 1].

The swarm in the original PSO to search the solutions space can be seen as the dynamic evolution of a dissipative system. In this system, particles' velocity is regarded as kinetic energy and the distance between the particles produces the potential energy. The algorithm converts the energy to the search behavior in solution space. After initialization of particles' position and velocity, the swarm's kinetic and potential energy have been identified. As the system's evolution, the algorithm converges and the system's energy decreases. The convergence is an important condition for an algorithm to work, but for complex problem, the algorithm could be trapped into a local optimum, i.e., prematurity, which prevent the algorithm from finding better solution. As thus, some heuristic algorithm, such as simulated annealing, adopts a strategy to slow down the convergence. However, in this paper, we introduce an updating mechanism to interrupt the premature convergence of the PSO. When the algorithm converges and the energy is blow a threshold, updating mechanism is triggered to stop the convergence. In this way, the IPSO achieves a kind of intermittent convergence.

The IPSO has four main extensions based on the standard PSO:

1. Define a variable E to measure the energy of the swarm.
2. Design an updating mechanism to restart the stagnated algorithm.
3. Set a threshold E_T of energy E. The threshold can be adapted according to the dynamic information obtained from the evolution of the algorithm.
4. Implement intermittent convergence of the algorithm by a switch between the converged PSO evolution and the updating mechanism.

1) The measurement of the potential energy

The swarm's energy is determined by its potential energy and kinetic energy. The potential energy is dependent on the historical best position (pbest) of each particle because the velocity is produced by the distance between a particle's current position and pbest's position (global best position-gbest is one of the pbest positions too). And the kinetic energy is captured by its velocities. Therefore, we define the energy of the swarm as follows:

$$E = \frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D}(pb_{id} - x_{md})^2 + \frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D} v_{id}^2 \quad (3)$$

Where $x_{md} = \frac{1}{D}\sum_{i=1}^{N} pb_{id}$ is the dth coordinate of the pbests' centre. The first term denotes the potential energy by the average square distances between each pbest's and the pbests' centre. The second term is the average square velocities of particles representing the kinetic energy. As the E decreasing, the searching energy is reduced, so the

system requires external energy to be injected. Thus, an updating mechanism is adopted to increase the system energy.

2) Updating mechanism

Since the potential energy is dependent on the pbests, the pbests are renewed in updating mechanism. The updating rule is: firstly, one dimension denoted by k of pbest i is chosen randomly, and then a uniform distributed velocity is added to this chosen dimension. Specifically, the velocity added to pbest i denoted by v_{ik} is

$$v_{ik} = V\Delta rand \,. \tag{4}$$

Where Δ conforms to Bernouli distribution which takes the values ± 1 with probability 1/2, *rand* is a random number conforming to the [0,1] uniform distribution, and V is linearly decreasing from X_{max} to $0.1X_{max}$, where $[-X_{max}, X_{max}]$ is defined to the feasible bound of solution. The updating rule of the pbests are

$$pb_{ik} = pb_{ik} + v_{ik} \,, i=1,\ldots,n. \tag{5}$$

This updating mechanism only changes one dimension to update, so other dimensions' information of pbests is saved. In this way, both updated information and good memories are used to direct the particles behavior, which increases the efficiency of optimization.

3) The velocity formula of the improved PSO

In the original PSO, the inertia velocity term keep the particle move in the former direction which partly prevents the algorithm from prematurity. Since the updating mechanism in the new proposed algorithm plays the role of avoiding trapped into the local optimum, the inertia item is eliminated in order to speed up convergence. So we use the velocity formula

$$v_{id} = \gamma(c_1 rand_1(pg_{id} - x_{id}) + c_2 rand_2(pg_d - x_{id})) \,. \tag{6}$$

Where γ is constriction factor which can prevent the algorithm from divergence[5].

4) Adaptive threshold

It is important to set the value of threshold E_T because E_T decides when to run PSO evolution and when to run updating mechanism. The algorithm begins with $E>E_T$ and performs PSO evolution according to equ. (6). During the iterations, the algorithm converges accompanied by the decrease in energy. When $E \le E_T$, the algorithm starts the updating mechanism followed by an increase of E. When $E>E_T$, the algorithm returns to the PSO evolution.

E_T is dynamically adjusted according to the changes of fitness. When the improvement of fitness f is less than a small number ε within Ps iterations, i.e.,

$$\Delta f = \left| \frac{f_s - f_{s+Ps}}{f_s} \right| < \varepsilon \cdot \tag{7}$$

the threshold is adjusted to the current value of energy i.e., let $E_T = E$. In this case, the algorithm runs in a high energy state and allows a large scale search by updating the pbests. During the evolution of the algorithm, the threshold E_T is decreasing by ΔE_T per iteration.

$$\Delta E_T = (E_T - E_{T\min}) / (Maxstep - s) \,. \tag{8}$$

Where s is the current iteration number and *Maxstep* denotes the max iteration number. Equ. (8) can guarantee the algorithm ends at E_{Tmin} which is a desired energy threshold when the algorithm terminates. The switch process between PSO evolution and updating is described as Fig. 1. We define an upper bound upE_T for E to avoid the algorithm from divergence due to continued update.

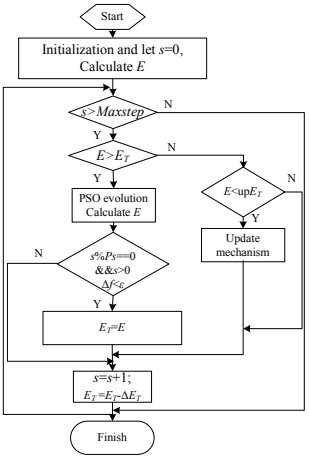


Fig. 1. Switch process between PSO evolution and updating mechanism

3 Numerical Experiments and Results

In order to demonstrate the performance of the proposed PSO, we use 10 benchmark functions defined in literature [13] for the experimental tests. Since the biased initialization can test the algorithm global explore ability to expand its search beyond its initial limits. Therefore, we define the initialization range for all the 10 functions are $[X_{max}/2, X_{max}]$. This setting increases the difficulty of searching the optimum. The numerical experiments show that biased initialization has the most impact for function f8 whose landscape declines steeply around the global optimum (the center of the feasible space) and while declines flatly in the region far away from the global optimum. So it is quite possible for algorithms to be trapped in local optima In order to compare the difference of the symmetry and the biased initialization, let f8 denote the initialization range $[-X_{max}, X_{max}]$ and f8' denote the initialization range $[X_{max}/2, X_{max}]$ in which the function value declines very flatly and there are a lot of local optima. The experiment results of the two initialization range are listed in table 1.

We compare other 4 PSO variants (gbest, lbest, FIPS and HPSOTVAC) with the proposed IPSO. Note that all the PSO algorithms use the same population size of 30, the same number of 300000 fitness evaluations (FEs) and the same maximum velocity $V_{max} = X_{max}$ (half of the feasible space) in experiments. Gbest, lbest optimization mechanism use the constricted PSO's updating rules. In FIPS, the von Neumann topology structure is applied and the parameter φ is set to 4.1. HPSOTVAC uses the parameters value defined in the literature [6].The parameters of IPSO are set as follows:

c_1=2.05, c_2=2.05, $\gamma = 0.72984$, $E_T = 10^{-4} X_{\max}^2$, $upE_T = X_{\max}^2$ and Ps=200. The experimental results, in terms of the mean and standard deviation of the fitness value obtained from 20 independent trials, are displayed in table 1.

The best results obtained by the 5 PSO algorithms are shown in bold. The results indicate that the IPSO achieves the best results on most complex multimodal functions f6, f7, f8', f9, f10, especially on f8' where all other PSOs experimented in this paper fail to jump out of the local optima far away from the global optimum. And for complex function f6 and f7, the IPSO also performs much better when comparing with other PSOs.

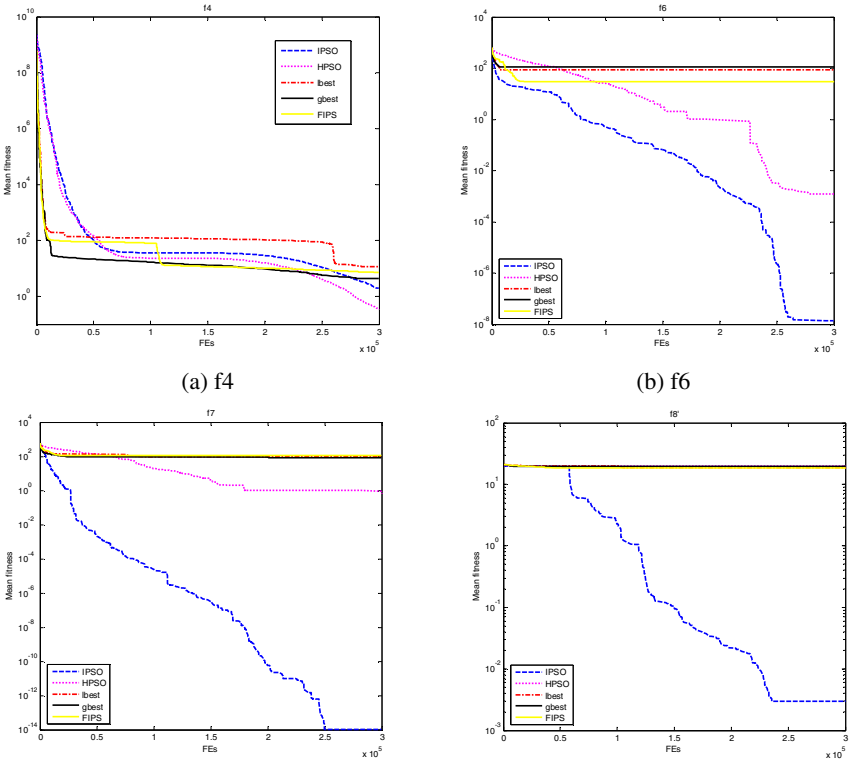
Table 1. Search results comparisons among different algorithms

function		gbest	lbest	FIPS	HPSOT-VAC	IPSO
Sphere	Mean	**1.5E-180**	7.1E-90	1.27E-32	2.3E-51	1.3E-28
f_1	Dev	0	0	3.2E-32	4.4E-52	2.2E-25
Quadric	Mean	1.3E-2	0.204	0.176	1.2E-3	**6.1E-7**
f_2	Dev	2.0E-2	0.14	0.128	3.6E-3	6.5E-6
Schwefel2.22	Mean	1.4E-61	9.4E-53	**1.32E-62**	2.2E-20	3.2E-48
f_3	Dev	6.1E-61	2.5E-52	6.23E-63	9.0E-21	4.6E-48
Rosenbrock	Mean	4.58	11.6	24.53	**0.248**	2.68
f_4	Dev	5.53	13.2	12.33	0.578	1.95
Griewank	Mean	0.032	0.017	**8.91E-6**	5.9E-3	0.014
f_5	Dev	0.026	0.015	3.27E-3	9.4E-3	0.027
Rastrigin1	Mean	109.4	93.4	30.2	3.7E-3	**3.0E-7**
f_6	Dev	26.4	20.2	8.8	5.4E-2	7.3E-6
Rastrigin2	Mean	90.45	97.6	110.00	0.61	**1.1E-14**
f_7	Dev	20.95	25.1	30.82	0.68	2.8E-13
Ackley	Mean	1.69	3.1E-3	**7.9E-15**	3.5E-7	9.3E-14
f_8	Dev	1.10	0.17	8.3E-15	9.7E-8	7.5E-14
Ackley	Mean	19.11	18.6	18.92	17.91	**3.1E-3**
f_8'	Dev	5.32	8.01	7.45	5.09	2.4E-3
Penalized P8	Mean	0.364	6.9E-15	9.9E-29	1.8E-14	**9.9E-32**
f_9	Dev	0.453	5.7E-14	6.2E-29	7.4E-13	3.1E-31
Penalized 16	Mean	0.049	1.6E-3	5.3E-12	4E-14	**3.3E-28**
f_{10}	Dev	0.141	4.0E-3	7.64E-13	1.9E-14	5.3E-27

There is a cost for tuning the IPSO to obtain ability to jump out of the local optima, and the cost is the slow convergence on unimodal problems. Even so, IPSO has already obtained good results which is good enough to be applied in practice.

The fitness descent characteristics of the evolution processes are shown in Fig. 2 (f4, f6, f7 and f8' selected for illustration). It is also observed from Fig. 2 the IPSO is capable of producing a quality of optimal solution with faster rate, while other PSOs such as gbest, lbest and FIPS stop optimizing in early stage of iterations. This is due to the updating mechanism which restarts optimization, so that the fitness will continue to be improvement.

In summary, the IPSO can exploit the solution space by PSO evolution and explore the space by updating mechanism. Fig. 2 also reveals that the IPSO generally offers a high speed of convergence, especially for the functions f6, f7 and f8' whose global optimum is difficult to be found.



(a) f4 (b) f6

(c) f7 (d) f8'

Fig. 2. Algorit5hm convergence for selected functions f4, f6, f7 and f8'

4 Conclusion

In this paper, we propose an improved PSO algorithm embedded an adaptive updating mechanism. As the PSO algorithm converges, the swarm's energy E decreases. When $E \leq E_T$ (E_T is dynamic adjusted during the iterations according to the variations of fitness), the updating mechanism begins to work which injects new energy into the swarm so as to increase E. When $E > E_T$, optimization via PSO evolution is resumed. By the switch between the PSO without inertia term which plays the role of local search and updating mechanism which can expand the explore range, the algorithm achieves a good balance between global exploration and local exploitation. In addition, the IPSO is easy to be implemented because only a little change is required based on the original PSO. Compared with gbest, lbest, FIPS and HPSOTVAC on benchmark functions, our approach has demonstrated good performance in terms of global search ability to jump out the local optima. In general, the IPSO offers the best accuracy on functions f2, f6, f7, f8', f9 and f10, especially for function f8' whose initialization range does not contain the global optimum, the solution found by the IPSO is much better than that found by the other 4 PSO algorithms.

Acknowledgments. This work is supported by the Natural Science Foundation of Shanghai (No. 09ZR1401800) and the Shanghai Education Development Foundation (No. 2008CG38).

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: 4th IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Press, Piscataway (1995)
2. Tasgetiren, M.F., Liang, Y.C., Sevkli, M., Gencyilmaz, G.: A Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in the Permutation Flowshop Sequencing Problem. European Journal of Operational Research 177, 1930–1947 (2007)
3. Franken, N., Engelbrecht, A.P.: Particle Swarm Optimization Approaches to Coevolve Strategies for the Iterated Prisoner's Dilemma. IEEE Trans. Evol. Comput. 9, 562–579 (2005)
4. Ho, S.Y., Lin, H.S., Liauh, W.H., Ho, S.J.: OPSO: Orthogonal Particle Swarm Optimization and Its Application to Task Assignment Problems. IEEE Trans. Syst., Man, Cybern. A, Syst., Humans. 38, 288–298 (2008)
5. Poli, R., Kennedy, J., Blackwell, T.: Particle Swarm Optimization: An Overview. Swarm Intelligence 1, 33–57 (2007)
6. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients. IEEE Trans. Evol. Comput. 8, 240–255 (2004)
7. Oca, M.A.M., Stützle, T., Birattari, M., Dorigo, M.: Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm. IEEE Trans. Evol. Comput. 13, 1120–1132 (2009)
8. Zhan, Z.H., Zhang, J., Li, Y., Chung, H.S.H.: Adaptive Particle Swarm Optimization. IEEE Trans. Syst., Man, Cybern. B, Cybern. 39, 1362–1380 (2009)
9. Chatterjee, A., Siarry, P.: Nonlinear Iinertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization. Comput. Oper. Res. 33, 85–871 (2004)
10. Tripathi, P.K., Bandyopadhyay, S., Pal, S.K.: Adaptive Multi-Objective Particle Swarm Optimization Algorithm. In: Proc. IEEE Congr. Evol. Comput., Singapore, pp. 228–2288 (2007)
11. Kennedy, J., Mendes, R.: Population Structure and Particle Swarm Performance. In: Proc. 2002 Congress Evolutionary Computation, vol. 2, pp. 1671–1676 (2002)
12. Lovbjerg, M., Krink, T.: Extending Particle Swarm Optimizers with Self-Organized Criticality. In: Proc. Congr. Evol. Comput., Honolulu, HI, pp. 1588–1593 (2002)
13. Qi, J., Pang, S.: Re-Diversified Particle Swarm Optimization. In: Li, K., Fei, M., Jia, L., Irwin, G.W. (eds.) LSMS 2010 and ICSEE 2010, Part II. LNCS, vol. 6329, pp. 30–39. Springer, Heidelberg (2010)

Mortal Particles: Particle Swarm Optimization with Life Span

Yong-wei Zhang, Lei Wang, and Qi-di Wu

College of Electronics and Information Engineering
Tongji University, Cao'an Road 4800
201804, Shanghai, P.R. China
yongwzhang@gmail.com

Abstract. Born and death is the nature of lives, but most swarm intelligence algorithm did not reflect this important property. Based on Particle Swarm Optimization, the concept of life span is introduced to control the activity generation of particles. Furthermore, the differential operator is applied to enhance the convergence and precision. The performance of propose algorithm, along with PSO and DE, is tested on benchmark functions. Results show that life span and differential operator greatly improved PSO and with well-balanced exploration and exploitation characteristic.

Keywords: Swarm Intelligence, PSO, Life Span, Differential, Mutation, Optimization, Algorithm.

1 Introduction

A branch of artificial intelligence which deals with the collective behaviour of swarms through complex interaction of individuals without supervision, is referred to as swarm intelligence [1]. During past decades, many swarm-based optimization algorithms have been developed and put into dealing with real world problems. Ant Colony Optimization (ACO) [2,3]mimics the foraging behavior of ant colonies, which uses the pheromone to deliver the landscape information of target problem between ants. Particle Swarm Optimization (PSO) [4] imitates the social behavior of flock of birds or school of fishes, and the particles share their experiences through a component of velocity update formula. Artificial Bees Colony (ABC) [5] utilizes the divided labor of bees to explore and exploit the nectar resources, and the message about food source is represented by the dances of employee bees. The Differential Evolution (DE) [6], even not simulate any real-world system, is still considered as a swarm intelligence algorithm because of the interaction among individuals.

As an important part of swarm intelligence algorithms, PSO was broadly studied and applied to many engineering problems such as controller design [7,8], task assignment problem [9], scheduling [10], etc. Many variants of PSO

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 138–146, 2011.
© Springer-Verlag Berlin Heidelberg 2011

are designed as well, such as PSO with inertia weight[11], PSO with linearly decreasing $V_{\max}$[12], PSO with collision-avoiding mechanism [13] etc..

Though PSO claims to be an algorithm that imitates the behavior of birds or fishes, the behavior of particles is more like a bunch of robots than living biomes. Except the initial phase, the particles never born or die. In other words, the particles are immortal. To the author's knowledge, few literatures mentioned the life span of particles, not along the effect the living particles will bring.

Recently, the differential operator, as a high-efficient heuristic, is increasingly introduced into various algorithms. For example, in Shuffled frog-leaping algorithm (SFLA)[14], the new solutions are generated based on the difference between current solution and the best solution of the memeplex. In the latest version of ABC [15], a differential operation is also used. The appropriate utilization of differential operator balances the exploration and exploitation of a search process. For PSO, there is no doubt that the introduction of differential operator will accelerate convergence and improve accuracy.

In our study, we introduce the concept of life span into the PSO scheme (mortal particles). To further improve the algorithm performance, the differential operator is used to generate candidate solutions out of the memory of individual particles. The performance of proposed algorithm, called Mortal Differential PSO (MDPSO), is compare with the original algorithms (immortal particles).

The rest of this paper is organized as follow. Section two briefly introduces the PSO algorithm. In section three, we propose the Mortal PSO scheme. The experiment results are presented in section four. Section five gives discussion and conclusion.

2 Particle Swarm Optimization

Since first introduced by Kennedy and Eberhart [4] in 1995, PSO has developed numerous variants. A detailed description of these variants of PSO can be found at [16]. Our study is focused on original PSO.

A D-dimension unconstrained minimization problem is defined as follow.

$$\min f(\boldsymbol{x}), \boldsymbol{x} = [x_1, \cdots, x_D] \tag{1}$$

where D is the dimension of the problem or the number of parameters to be optimized, $\mathbf{x}$ is a solution of the problem, $x_{i\min} \leq x_i \leq x_{i\max}$.

In Particle Swarm Optimization, the position of a particle represents a solution. And the motion of particle P_i is driven by velocity V_i. Through updating the velocity and the position of particles, the feasible space is searched. The updating formula of velocity and position are as follow.

$$V_i(t+1) = V_i(t) + c_1 rand_1(pbest_i - P_i) + c_2 rand_2(gbest - P_i) \tag{2}$$

$$P_i(t+1) = P_i(t) + V_i(t+1) \tag{3}$$

where $pbest_i$ is the best position previously discovered by particle P_i, $gbest$ is the best position discovered by the entire population. $rand_1$ and $rand_2$ are uniformly

distributed random number in region [0,1]. c_1 and c_2 are accelerating constants that reflect the weighting of stochastic acceleration term that pull particles to *pbest* and *gbest* position[16]. The range of P_i and V_i is limited in $[X_{\min}, X_{\max}]$ and $[V_{\min}, V_{\max}]$. When parameter exceeds the upper or lower bounds, we simply set parameter as the value of bounds. Some researcher use different random numbers in every dimension, as defined in (4)-(5).

$$V_i^d(t+1) = V_i^d(t) + c_1 rand_1^d(pbest_i^d - P_i^d) + c_2^d rand_2^d(gbest^d - P_i^d) \quad (4)$$

$$P_i^d(t+1) = P_i^d(t) + V_i^d(t+1) \quad (5)$$

where superscript d is the dimension index. According to (4), every component of velocity V_i needs newly generated random numbers weighting the portion of individual and global terms. The difference between (2)-(3) and (4)-(5) is reported in [17]. For clear discussion, we use (2)-(3) in our study.

3 Mortal Particle Swarm Optimization

3.1 Mortal Particles

Death, as one kind of updating mechanisms in biomes, is an indispensible part of evolution. To sustain the population with limited resources, the old and weak individuals have to make room for the young and fitter ones. In the context of swarm intelligence, the elimination of old solutions plays the same role with death in the real world. However, most swarm intelligence algorithms did not take the life span of solutions into account. From the beginning to the end of a search process, a solution is only changed, not been replaced. If the change stops, i.e., the solution trapped in local extreme, then the premature convergence is unavoidable unless some other operations are introduced.

Consider (2), when a solution is trapped, it means *gbest* and $pbest_i$ are remain unchanged. This solution can easily replicate itself and occupy the whole population. Because the *gbest* and $pbest_i$ will attract the rest particles to them, and eventually to *gbest*[16]. Since all the particles are immortal, the algorithm will end up with a population that has same individuals. To help the solution escape local, we use 'life span' to control the number of activity generations of a particle. For problem (1), the life span of a particle is decided by following formula.

$$life_i = \exp\left(\frac{\min_{i=1}^{N}(f_i) - f_i}{\sum_{i=1}^{N}(f_i - \min_{i=1}^{N}(f_i))/N}\right) \quad (6)$$

where f_i is the evaluation of object function, N is the number of particles.

The life span of each particle is within (0,1]. The life of a particle will reduce Δ with every position update. When $life_i \leq 0$, the $pbest_i$ particle will be eliminated and reinitialized within the search space. The $pbest_i$ is reinitialized as well to erase the memory of the particle, otherwise the new particle will be attracted to $pbest_i$ again.

In some cases, the evaluation of some particle are far bigger than the rests, thus leads to a large denominator in (6), and eventually shorten the life span of all particles. In that case, the following formula can be used instead.

$$life_i = \exp\left(\frac{\min_{i=1}^{N}(f_i) - f_i}{\text{median}(f_i - \min_{i=1}^{N}(f_i))/N}\right) \tag{7}$$

where $\text{median}(f_i - \min_{i=1}^{N}(f_i))$ is median number.

3.2 Differential Operator

With mortal particle, the algorithm can escape local now. But the convergence speed is still a problem. If the algorithm reinitializes the solutions again and again to explore wider region, it becomes random search. In addition, the life span did not necessarily speed up the convergence. In this context, we introduce differential operator to improve the convergence.

$$Pn_i^d = \begin{cases} pbest_i^d + (pbest_{r1}^d - pbest_{r2}^d), & \text{if } rand < p \\ pbest_i^d, & \text{else} \end{cases} \tag{8}$$

$$\begin{array}{c} r1,\ r2 \in [1, \cdots, N] \\ r1 \neq r2 \end{array} \tag{9}$$

where Pn_i^d is the d^{th} variable of i^{th} new particle, p is the mutation probability, $r1$ and $r2$ are random integers. From our observation, $p = 0.15$ to 0.25 is suitable for most problems.

The P_i and $pbest_i$ is updated according to following.

$$f_{\text{obj}}(Pn_i(t)) < f_{\text{obj}}(pbest_i) \longrightarrow P_i(t+1) = Pn_i(t),\ pbest_i = Pn_i(t) \tag{10}$$

where $f_{\text{obj}}(\cdot)$ is the object function.

The pseudo code of MDPSO see Algorithm 1. Where FE and $FE_{\max}$ denote the number of function evaluation and the maximum of it.

4 Validation Experiments

4.1 Parameter Setting

To verify the proposed algorithm, numerical experiments are conducted. The performance of MDPSO is compared with PSO and DE. The DE parameters are set according to [18]. The rest parameters are set as follow. Accelerating constants $c1 = c2 = 2$, population size $N = 20$, mutation probability $p = 0.15$, maximum number of function evaluation $FE_{\max}$=2e5, velocity V_i is limited by (11).

$$\begin{array}{c} V_{\max} = (X_{\max} - X_{\min})/10 \\ V_{\min} = -(X_{\max} - X_{\min})/10 \end{array} \tag{11}$$

Algorithm 1. Mortal Differential PSO

```
Object function f(x), x = (x_1, ..., x_D)^D
Generate N particles P_i(i = 2, 3, ... N), evaluate particles, record pbest_i and gbest
Compute life value of particles by (7)
while FE < FE_max do
  for each particle P_i do
    Update V_i and P_i by (2)-(3), FE = FE + 1, evaluate P_i : f_i = f_obj(P_i)
    if f_i < f_obj(pbest_i) then
      pbest_i = P_i
    else
      life_i = life_i − Δ
    end if
    if life_i < 0 then
      Reinitialize P_i and evaluate it, FE = FE + 1
    end if
  end for
  Generate new particles by (8)-(9) and evaluate them, FE = FE + N
  Update P_i and pbest_i according to (10)
  Update particle life by (7) and record gbest
end while
Post process results and visualization
```

4.2 Benchmark Functions

The benchmark functions are usually used as standard test bed for optimization algorithms. To avoid the unintentional attraction of zeros, the benchmark functions are shifted with a displacement $\boldsymbol{d} = (d_1, d_2, \cdots d_D)^D$ according to following formula.

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{x} - \boldsymbol{d} \\ f &= f_{\text{obj}}(\boldsymbol{z}) \end{aligned} \tag{12}$$

For example, the Non-continuous Rastrigin function has numerous local minima, which located at small flat local regions. The flat region has no differential information and the function is un-derivable within its search space, which makes this function very difficult for most search algorithms. The function's shifted version landscape is shown at Fig. 1. The benchmark functions used in our experiment are listed at Table 1. To test the overall performance of proposed algorithm, we try to cover all kinds of test functions. As the characteristic showed at Table 1, four functions are unimodal and four are multimodal. At the same time, four functions are separable and four are non-separable.

4.3 Simulation Results

The initial and final locations of 20 particles of 2-D Non-continuous Rastrigin function in MDPSO are shown at Fig. 2. The final locations are around the optimal solution [1.28,1.28], but distances between particles are still fairly large. This explains why MDPSO can escape local. To further demonstrate the

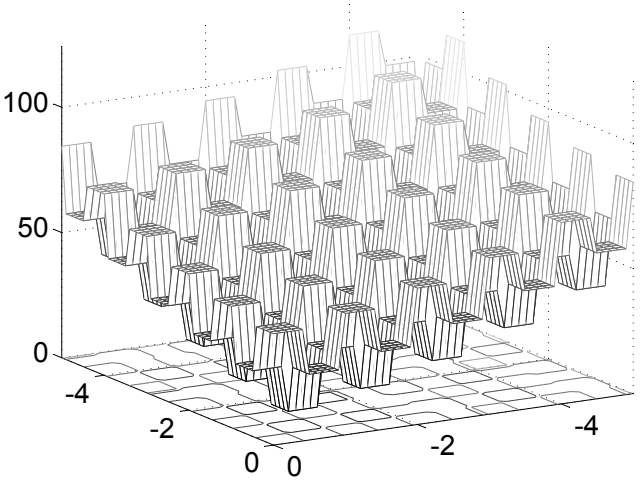


Fig. 1. The 3-D landscape of Shifted Non-continuous Rastrigin function

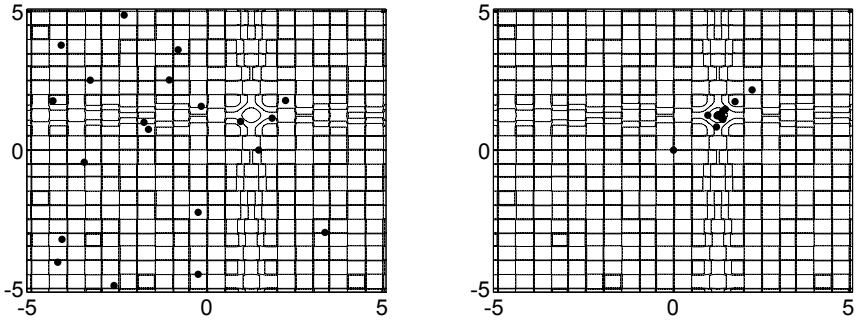
Table 1. Benchmark Functions. D: Dimension, F: Feasible Bounds, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable, R: Regular, I: Irregular.

Name	Equation	D	F	C	**d**
Sphere	$f_i = \sum_{i=1}^{D} x_i^2$	30	[-100, 100]	USR	25^D
Schwefel 2.21	$f_2 = \max_{i=1}^{D} \lvert x_i \rvert$	30	[-100, 100]	USI	25^D
Rosenbrock	$f_3 = \sum_{i=1}^{D} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	[-30, 30]	UNR	0^D
Schwefel 2.22	$f_4 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	[-10, 10]	UNI	2.5^D
Rastrigin	$f_5 = \sum_{i=1}^{D} \left[x_i^D - 10\cos(2\pi x_i) + 10\right]$	30	[-5.12, 5.12]	MSR	1.28^D
Non-continuous Rastrigin	$f_6 = \sum_{i=1}^{D} \left[y_i^D - 10\cos(2\pi y_i) + 10\right]$ $y_i = \begin{cases} x_i & \lvert x_i \rvert < \frac{1}{2} \\ \frac{\text{round}(2x_i)}{2} & \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	30	[-5.12, 5.12]	MSI	1.28^D
Griewank	$f_7 = \frac{1}{4000} \sum_{i=1}^{D} x_i^D - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600, 600]	MNR	150^D
Ackley	$f_8 = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i)\right) + 20 + \mathrm{e}$	30	[-32.768, 32,768]	MNR	8.192^D

effectiveness of proposed algorithm, 50 runs are conducted for each algorithm, and the results are summarized at Table 2. For each benchmark function, the best, median, worst, mean, standard deviate (Std) and success (Scr) rate of 50 runs are listed. The success rate is determined by (13)-(14).

$$\text{Scr} = N_{\text{success}}/\text{Maxrun} \times 100\% \tag{13}$$

$$|f_{\text{best}} - f_*| < \varepsilon \tag{14}$$



(a) Initial Locations (b) Final Locations

Fig. 2. Initial and final 2-D locations of 20 particles in MDPSO

where N_{success} is the number of success runs. If f_{best} satisfy (14), a success run is recorded. In our experiment, the tolerance is truncated at 1.00e-8, which means all values that less than 1.00e-8 are treated as zero. From Table 2 we can conclude that MDPOS outperforms PSO for all eight test functions and outperforms DE for three multimodal functions (Rastrigin, non-continuous Rastrigin and Griewank) and one unimodal function (Schwefel 2.21). DE outperforms MDPSO for two unimodal non-separable functions (Rosenbrock and Schwefel 2.22). Both MDPSO and DE have same performance for Sphere and Ackley function.

Table 2. Comparison of Three Algorithms

	Function	f_1	f_2	f_3	f_4	f_6	f_6	f_7	f_8
	ε	1.00e-8	0.5	25.0	1e-8	1e-8	1e-8	1e-8	1e-8
PSO	Best	0.07	0.78	29.04	0.16	33.91	38.00	0.22	0.10
	Median	0.19	1.41	74.15	0.27	63.73	69.00	0.40	0.22
	Worst	0.38	3.29	368.86	0.46	94.62	126.00	0.64	1.71
	Mean	0.19	1.60	87.93	0.27	63.88	71.88	0.41	0.53
	Std	0.07	0.57	70.88	0.07	14.61	16.96	0.09	0.50
	Scr	0%	0%	0%	0%	0%	0%	0%	0%
DE	Best	0.00	0.06	0.04	0.00	15.69	32.67	0.0	0.0
	Median	0.00	2.24	17.40	0.00	83.48	81.43	0.0	0.0
	Worst	0.00	20.66	69.43	0.00	133.97	118.62	7.40e-3	0.0
	Mean	0.00	3.44	26.12	0.00	84.25	81.25	1.48e-4	0.0
	Std	0.00	4.02	21.63	0.00	27.56	22.97	0.001	0.0
	Scr	100%	16%	78%	100%	0%	0%	98%	100%
MDPSO	Best	0.00	0.17	4.80e-3	0.00	0.00	0.00	0.00	0.00
	Median	0.00	0.28	20.78	0.00	0.00	0.00	0.00	0.00
	Worst	0.00	0.41	81.90	12.50	5.59e-7	0.00	0.00	0.00
	Mean	0.00	0.29	33.19	0.25	1.16e-8	0.00	0.00	0.00
	Std	0.00	0.05	27.92	1.75	7.82e-8	0.00	0.00	0.00
	Scr	100%	100%	72%	98%	96%	100%	100%	100%

Clearly MDPSO performed well when deals with multimodal functions. Especially for Rastrigin and non-continuous Rastrigin function, MDPSO is far more superior. On the other hand, the advantage of DE is not obvious. In general, the proposed algorithm is more efficient than PSO and DE.

5 Discussion and Conclusion

From the experiment results we can conclude that the life span and differential operator significantly improved the performance of PSO and showed great potential on dealing with multimodal problems. With life span, some old particles are eliminated, so is its memory, *pbest*. With the elimination of old particles, the population diversity is increased. Technically the population will not converge to a same local extreme (like the original PSO always do), hence the algorithm will not trap. Theoretically, as long as the algorithm keeps running, it will find the global minima eventually.

Though the algorithm will not trap in local, it is not practical if it takes long time to converge. A differential operator can speed up convergence and increase the accuracy. In the early stage of searching process, the *pbest* records only the information of scattered local regions, and the average distance of particles is large. The new generated particle can search broader region. In the later stage, with the decreasing distance of particles, the differential factor becomes smaller. This drives the algorithm to higher precision. The process is self-adaptive, the exploration and exploitation are well balanced.

The implementation of life span and differential operator dose not depend on algorithms, which means the proposed improvement can be applied to other algorithms as well. PSO is just an example, how to improve other algorithms by these heuristics is still an open issue.

Acknowledgement. The research is supported by Ph.D. Programs Foundation of Ministry of Education of China (No. 20100072110038), National Natural Science Foundation of China (NSFC, No. 70871091, 61075064, 61034004), Foundation of the Ministry of Education of China for Returned Scholars and Program for New Century Excellent Talents in University of Ministry of Education of China.

References

1. Akay, B., Karaboga, D.: A modified Artificial Bee Colony algorithm for real-parameter optimization. Inform. Science (in press) (2010), Corrected Proof doi:10.1016/j.ins.2010.07.015
2. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of Cooperating agents. IEEE Trans. Syst. Man. Cybern. Part B. Cybern. 26, 29–41 (1996)
3. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. 1, 28–39 (2006)

4. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
5. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report, Computer Engineering Department, Engineering Faculty, Erciyes University (2005)
6. Storn, R., Price, K.: Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute (1995)
7. Zamani, M., Sadati, N., Ghartemani, M.K.: Design of an H PID controller using Particle Swarm Optimization. Int. J. Contr. Autom. Syst. 7, 273–280 (2009)
8. Zhang, Y., Qiao, F., Lu, J., Wang, L., Wu, Q.: Performance Criteria Research on PSO-PID Control Systems. In: 2010 International Conference on Intelligent Computing and Cognitive Informatics (ICICCI), pp. 316–320 (2010)
9. Salman, A., Ahmad, I., Al-Madani, S.: Particle swarm optimization for task assignment problem. Microprocess. Microsy. 26, 363–371 (2002)
10. Bo, L., Ling, W., Yi-Hui, J.: An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling. IEEE Trans. Syst. Man. Cybern. Part B. Cybern. 37, 18–27 (2007)
11. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proc. of the IEEE Int'l Conf. of Evolutionary Computation, pp. 69–73. IEEE Press, Piscataway (1998)
12. Fan, H.Y., Shi, Y.: Study on Vmax of particle swarm optimization. In: Workshop Particle Swarm Optimization (2001)
13. Blackwell, T.M., Bentley, P.: Don't push me! Collision-avoiding swarms. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 1691–1696 (2002)
14. Eusuff, M.M., Pasha, K.L.F.: Shuffled frog-leaping algorithm: a memetic metaheuristic for discrete optimization. Eng. Optimiz. 38, 129–154 (2006)
15. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global. Optim. 39, 459–471 (2007)
16. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. Ieee T. Evolut. Comput. 10, 281–295 (2006)
17. Wilke, D.N.: Analysis of the particle swarm optimization algorithm. Dept. Mechanical and Aeronautical Eng., Univ. of Pretoria, Pretoria, South Africa, (2005)
18. Pedersen, M.E.H.: Good Parameters for Differential Evolution. Technical report, Hvass Computer Science Laboratories (2010)

PSO Based Pseudo Dynamic Method for Automated Test Case Generation Using Interpreter

Surender Singh Dahiya[1], Jitender Kumar Chhabra[1], and Shakti Kumar[2]

[1] National Institute of Technology, Kurukshetra, India
[2] Institute of Science & Technology, Kalawad, Yamunanagar, India
{surendahiya,jitenderchhabra,shaktik}@gmail.com

Abstract. In this paper, we propose a particle swarm optimization (PSO) based hybrid testing technique named as "pseudo dynamic testing" to generate test data for C programs by fulfilling one of the most demanding test adequacy criteria: the all-path testing criterion using an interpreter. The proposed methodology attempts to solve many of the structural testing problems such as dynamic variables, input dependent array index, abstract function calls, infeasible paths and loop handling. The key algorithms and heuristics are given which are easy enough to implement, scalable and effective. The technique is employed on real world programs to show the robustness of this technique. The set of test inputs generated are not redundant as each leads to a different program path.

Keywords: Pseudo dynamicSoftware Testing, Symbolic Testing, Dynamic Testing, Particle Swarm Optimization (PSO).

1 Introduction

The manual selection of test inputs to the program under test is both labor-intensive, and erroneous [1]. Automated test data generation can reduce the test cost and time significantly. Although automation in testing is less intelligent but it helps in avoiding biases in test process. For automatic test data generation in structural testing, there are two ways through which the program can be analyzed. The dynamic analysis approach can handle dynamic arrays, pointers, internal variables, input dependent array index, function calls and other dynamic constructs more accurately than static analysis approach such as symbolic testing but it may also be more expensive, since the program under test is executed repeatedly for searching the input data. This problem becomes worse, when the tester fails to apply any highly effective search algorithm for generation of values of the inputs to execute the desired path [10]. Another problem with dynamic testing is its uncontrolled execution behavior, where any infinite loop execution or an infeasible target path (a path selected for testing and no input combination exists to execute it) can make testing extremely slow. Therefore, the dynamic execution method is very costly and time consuming, making it inappropriate and impractical for nontrivial programs. On the other side, second method namely the symbolic testing provides a controlled environment for testing because, here, the tester can specify in advance the execution path. Some of the infeasible paths can be avoided by excluding them in the list of test paths by following some

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 147–156, 2011.
© Springer-Verlag Berlin Heidelberg 2011

identification processes such as identical or complement decisions, mutually-exclusive decisions prescribed by [7] and [11]. For other difficult-to-identify infeasible paths, the tester can monitor the search progress of test data generation process and if he/she feels that search is not making any progress from the last several iterations then the path can be declared infeasible or difficult to be covered for test case generation purpose. This also can be achieved automatically [2]. Although symbolic testing provides partial solution to the problem of infeasible paths but it miserably fails to generate test data for such programs where state of any variable is only determined at run time [3].

A careful study of both techniques can reveal that although both of the testing techniques have their own inherent incapability in solving testing issues if used in isolation but these complement each other. While symbolic testing is not able to handle dynamic variables, dynamic execution based method can efficiently test programs involving these types of variables. On the other side, while dynamic execution testing is not able to solve the problem of path explosion due to presence of loops in path traversal, symbolic execution provides the facility of restricted execution by virtue of its execution style. If these complement each other, a question may arise; why not to create a hybrid testing approach which imitates the behavior of both of the testing types? This paper presents a novel hybrid approach for automatic test data generation named as pseudo dynamic execution (PDE) testing to address most of the problems of structural testing techniques. The Pseudo dynamic testing is a well-accepted concept in civil and mechanical engineering where actual construction or product is not tested but a simulation test is conducted which simulates the actual environment of the product. In the proposed approach, we have exploited the same concept, where program's statements are executed concretely using an interpreter as done in dynamic approach but in a controlled way as done in the symbolic technique. This is the very reason to name this testing technique as pseudo dynamic technique. A soft computing based search algorithm particle swarm optimization (PSO) is also used to search for test data to achieve the all-path testing criterion. The hybrid approach can be used in test data generation for all type of real programs with/without loops and procedures.

The rest of the paper is organized as follows: Section 2 explains proposed methodology. Section 3 describes the key algorithms used in PSO based PDE method of test case generation. Section 4 provides an experimental evaluation of our technique on real-world programs. Section 5 compares the proposed methodology with existing similar methods. Section 5 isfollowed by the conclusion.

2 The Pseudo Dynamic Execution (PDE) Method

In the symbolic execution method, for each path, a path constraint is maintained together with a table which contains the expression of each variable referenced. When an assignment is encountered, the variables in the assignment statement are substituted by their current expressions maintained in the table and the resulting assignment expression replaces the expression representing the target variable. When a predicate is encountered, each variable in the predicate is substituted by its current expression and it is integrated with the path constraint. Once the symbolic execution of the path is complete, the path constraint is evaluated for its satisfaction by replacing symbols

representing variables by the concrete values derived from a constraint solver. The values of input variables, which satisfythe path-constraint, become a test datum. This method has one problem; if variables maintained in the table are of dynamic nature, whose expression can't be determined statically and depends on the concrete values then variables expression can't be substituted conclusively for predicate evaluation.

Table 1. Fitness function of a branch predicate

Violated predicate	Penalty to be imposed in case predicate is not satisfied
A < B	A – B + ζ
A <= B	A – B
A > B	B – A + ζ
A >= B	B – A
A = B	Abs(A – B)
A ≠ B	ζ – abs(A – B)
A and B are operands and ζ is a smallest constant of operands' universal domains used for incremental penalty in case of inequality violated predicate when both operands have equal value. For integer it is 1 and in case of real values it can be 0.1 or 0.01 depending on the accuracy we need in solution.	

We have tried to solve this problem by executing the statement of programs on the path for which test data is to be generated, one by one by using concrete values of input variables with the help of an interpreter, thus, removing the urgency of maintaining a table. Each variable is maintained in the program memory space by interpreter itself then updated accordingly as the execution of the statements progresses. If any predicate statement is encountered during the path execution progress then it is evaluated for its truthfulness based on the current program variable values and subsequently, a penalty value replaces corresponding predicate in the path constraint which is built by concatenating all the branch nodes predicates. This penalty value is calculated using the well-established branch distance rules concept [17] as shown in table 1. After completing the processing of the whole path using the above concept, each 'and' operator in path constraint is replaced with '+' operator, each 'or' operator is replaced with ',' operator and each '(' operator with 'min(' operator in path constraint string. Finally, a single penalty value is calculated by evaluating the whole path constraint string. If the penalty value is zero then the path constraint is satisfied and it implies that the concrete values of the inputs are able to execute the target path and this becomes the test data for the path. If the penalty value comes out to be more than zero then it means that the concrete values set for input variables fails to execute the target path fully and these can't form the valid test data, however, in a population based search algorithm such as PSO, these can provide direction for evolving such values which can execute the target path and generate the valid test data. An algorithm explaining the above concept is given in figure 1. Thus this methodology is an amalgamation of symbolic and concrete execution and is a dynamic modeling of program execution in a controlled environment just like as it is done in pseudo dynamic testing in other engineering domains.

```
For each feasible path
  For each particle in PSO search algorithm population
    Assign input variables values from PSO particle
    For each node in target path of CFG
      If node is non-branch node
        Then concretely execute all the statements related to that node using Ch
          interpreter;
      Else
        Find the predicate of branch node from node expression array.
        Find the traversal link to next node in target path from CFG matrix.
        If traversal link is for false execution of branch predicate
          Then simplify predicate for negation.
        End If
        If simplified predicate is evaluated true using Ch interpreter
          Then continue without any penalty to individual solution
        Else
          Extract distinct predicates from combined branch predicate.
          For each distinct predicate
            If distinct predicate is evaluated true by individual
              Then assign zero penalty corresponding to that distinct predicate
            Else
              Determine Penalty for violated distinct predicate by following the
                concept for branch distance function
            End If
          End For
          Replace each distinct predicate with its corresponding penalty in
            composite predicate.
        End If
        Replace each '&&' symbol with plus (+) sign and '||' symbol with comma (,)
        sign and each bracket '(' with min([ in branch predicate to determine fitness
        related to the combined node predicate.
      End If
      Add fitness of each branch node predicate to determine the fitness for individual
        related to whole path.
    End For
    If fitness for the path is zero
      Then record the individual as a test data for the particular path and exit inner
        loop.
    End If
  End For
End For
```

Fig. 1. Pseudo dynamic testing using PSO algorithm

3 PSO Based Test Data Generator

In the proposed methodology, test data is generated in two steps; first the program under test is instrumented to extract control flow graph (CFG) and node statements

array. All paths of the program are automatically generated from the CFG and some of the infeasible paths are deleted manually by identifying mutually-exclusive-decision on node predicates. Other infeasible paths, which are hard to be identified can be later resolved by considering the search progress of search algorithm[2]. Feasible paths and node expression are fed into the PSO search algorithm to generate test cases as shown in figure 2. The PSO search algorithm is implemented in 'Ch' interpreter environment [18] which is helpful in executing individual statements with the help of '*streval*' command. This command is also used to evaluate predicates which are further used to evaluate the whole path constraints which subsequently determine the fitness of an individual or a particle in PSO algorithm.

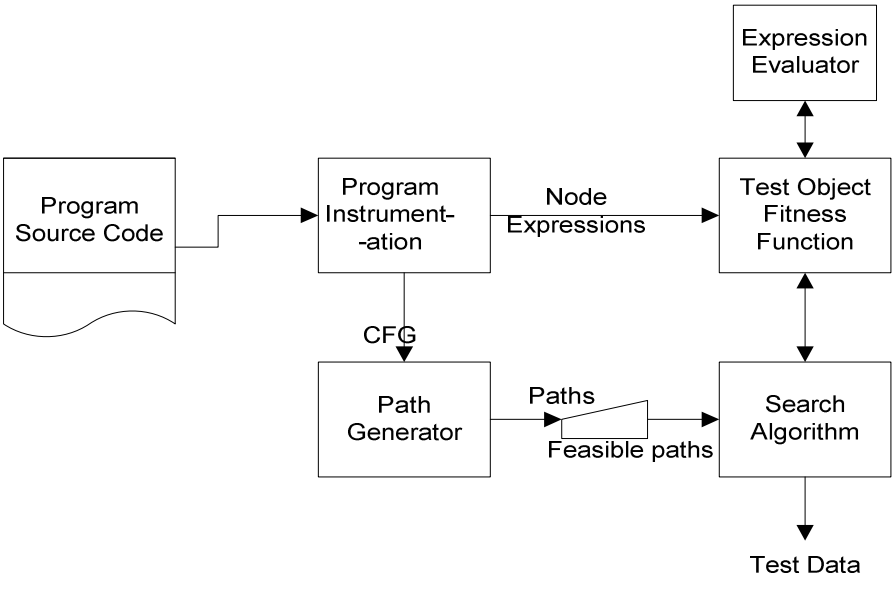


Fig. 2. PSO based Test data generator

The PSO search algorithm has the goal of covering all the paths in CFG of the program under test. The PSO generates tests (candidate solutions) and executes them as input for the program under test. A testis formed by a vector of given values $(y_1, y_2, \ldots, y_n)$ generated by the PSOfor the input variables $(z_1, z_2, \ldots, z_n)$ of the program under test. The general PSO algorithm appears in Fig 3. The PSO generates test inputs based on the test that is the current solution (CS). Initially, the CS is a random test, but, inside the loop, the PSO modifies it based on the cognition and social learning it has acquired from the previous solutions. When a test is generated, the PSO evaluates its fitness based on the extent of test criterion it is able to satisfy.

1. Initialize the particle population (Candidate Solution) by randomly assigning locations (X-vector for each particle) from input space of the program and velocities (V-vector with random or zero velocities- in our case it is initialized with zero vector)
2. Evaluate the fitness of the individual particle and record the best fitness P_{best} for each particle and update P-vector related to each P_{best}.
3. Also find out the individuals' highest fitness G_{best} and record corresponding position P_g.
4. Modify velocities based on P_{best} and G_{best} position using following equations
   4.1 $v_i^d(t+1) = w \times v_i^d(t) + \varphi_1 \times rnd() \times (p_i^d - x_i^d(t)) + \varphi_2 \times rnd() \times (p_g^d - x_i^d(t))$
   4.2 $x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$
   where
   - i is a particle.
   - d is dimension of the particle.
   - $rnd(\,)$ is random number generator.
   - φ_1 and φ_2 are are learning rates governing the cognition and social components.
   - g represents the index of particle with best p-fitness.
   - $t+1$ signifies next generation.
   - w is the inertia factor that dynamically adjust the velocities of particles gradually focusing the PSO into a local search.
5. Update the particles position using above equation.
6. Terminate if the condition is met
7. Go to Step 2

Fig. 3. PSO Algorithm

3 Experiments and Results

Ch integrated development environment (ChIDE) is used for implementing algorithms for experiment purpose. It is an interpreter and can execute ANSI standard C programs. Ten attempts are made for each path in a CFG of program under test for the purpose of averaging results. In each attempt, the PSO is iterated for 100 generations. The population size is 30. If a solution is not found within all attempts then, it is assumed that either the path is infeasible or the path constraint is too difficult to be solved. We have measured average percentage coverage (APC) of feasible paths, which measures the effectiveness of test data generator and is calculated as fraction of paths covered. The high value of APC is desirable. The starting values of inertia weight w, Cognition learning rate φ_1 and Social learning rate φ_2 are taken as 0.8, 2.8 and 1.3 respectively as suggested in [19]. For experiment, ten 'C' programs have been selected (henceforth called testing objects). Although, test objects are not very complex but they exhibit every characteristics for which we want to generate test cases. Except the first, each program contains loops. A2F & BS are the programs in which there are array indexes which depend on concrete values and can be solved only using dynamic analysis. Further there is a program linked list (LL1) which contains pointer variable and all feasible paths are covered for this object. A second version of link list (LL2) is also experimented where several abstract functions are taken and we are able to get result for the higher level function by implicitly and dynamically executing lower function.

The main attributes of these are objects and corresponding APC are shown in the table 2. From the results it can be concluded that this hybrid approach is effective on almost all programs. The APC for BS test object is just 58%. This is due to its requirement of a sorted array as input which destroy the search guidance for the PSO in every iteration and thus it becomes difficult to generate test cases especially where boundary cases are to be satisfied. Thus, it is a problem of search function not of the proposed testing technique.

Table 2. Test objects, their attributes and experimental result

Test Object	Lineof Code	Decision Nodes	Nesting Level	Loop exists	Pointer structure	Abstract-Function	Total Paths in CFG	Feasible Paths	APC %
TC	35	06	05	No	No	No	07	07	100
A2F	48	14	07	Yes	No	No	910	568	100
BS	23	04	03	Yes	No	No	124	62	58
REM	35	8	04	Yes	No	No	22	22	100
BUB	21	03	03	Yes	No	No	121	31	100
QUAD	24	05	03	Yes	No	No	06	06	100
MINMAX	27	03	03	Yes	No	No	121	121	100
ISPRIME	16	02	02	Yes	No	No	10	08	100
LL1	116	12	06	Yes	Yes	No	673	35	100
LL2	57	5	02	Yes	Yes	Yes	32	32	100

4 Related Work

Recently, researchers have started exploring the use of hybrid techniques for solving several issues of symbolic execution such as the use of dynamically allocated data and scalability. Godefroid used both types of testing strategies; concrete as well as symbolic (called concolic testing) in *Directed Automated Random Testing* (DART) [5] and *Systematic Modular Automated Random Testing* (SMART) [6] algorithms for test case generation of *C* programs.Sen *et al* [14] suggested a concolic unit test generation framework "CUTE" which is similar to the DART but it uses a logical input mapping table for handling pointer variables also. Majumdar and Sen [9] enhanced the coverage capability of CUTE by including random testing with concolic testing. Khurshid *et al* [8] proposed a new technique called generalized symbolic execution for automated test case generation for programs containing dynamic allocated data. Visser *et al* [15] combined model checking with symbolic execution to generate test inputs in Java programs. Deng *et al* [4] proposed *KUnit*, a unit test data generation framework for sequential, heap-manipulating Java applications. Pasareanu *et al* [12] used the symbolic execution extension to the Java Path Finder model checker to generate test cases for complex, safety-critical software.Saxena *et al* [13] introduced loop-extended symbolic execution (LESE) by including symbolic variables for the number of times each loop executes. Visvanathan and Gupta [16] generated test inputs for functions that requires pointer as input. All these techniques addressed testing requirements partially and were specific in their approaches and thus failed to address problems on larger scale.

Table 3. Comparison of different hybrid techniques

Criterion	DART	CUTE	PDE
Primary testing technique	Symbolic	Symbolic	Dynamic
Secondary technique	Concrete	Concrete	Not Required
Type of search algorithm	Random	Random	PSO based
Criterion for testing	Systematic Path testing	Systematic Path testing	All Path testing
Type of constraint solver	Linear Programming (LP)	LP with backtracking	PSO
Identification of feasible paths	Automatic with limited iteration based criterion	Automatic with limited iteration based criterion	Fully automatic with search progress criterion
Handling of abstract function	No	No	Yes
Handling of pointers	No	Pointer approximation only	Yes
non-linear constraints	No	No	Yes
memory mapping	Required	Required	Not Required

The proposed PDE based approach is most closely related to DART and CUTE. The DART starts with a random (concrete) input and collects symbolic path constraints (conditions) during its execution and then uses these to attempt direct execution down an unexplored path on the next execution by negating a predicate in the path condition. Thus a new path condition is solved, generating the new set of test inputs. When execution reaches a branching statement the underlying decision procedure cannot decide, the symbolic condition is replaced by its concrete value. Randomization is also used when automated reasoning is not possible. The DART handles constraints only on integer types, but does not handle constraints on the programs with pointers or complex data structures. For these cases, it only uses randomly generated data. The DART is also being extended into the SMART to perform inter-procedural static analysis for dynamic test generation at component level and defined test results as function summaries using

input pre-conditions and output post-conditions for reusing these for testing higher-level functions. The SMART algorithm takes a top-down, demand-driven approach to compute summaries in order to avoid explosion of paths in a large program but, it does not have any extenuating effects on the inherent limitations of symbolic execution. It fails to generate test cases in three circumstances; when in any program, call flow graph (inter procedural graph) is cyclic (as in the case of recursive calls of a function), second where any function for which pre-conditions summaries cannot be determined in advance (e.g. in functions where loop execution determine the behavior of specific summary of a function and its execution bound can't be determined statically) and lastly whenever a constraint on some inputs cannot be expressed within given theory of constraints (e.g if about a method it is not possible to reason completely by using information in the calling context to constrain the input values). The CUTE is similar to the DART with the additional capability of addressing pointer variables by explicit memory mapping. Table 3 compares the proposed method with CUTE and DART andbased upon the literature survey, theoretically, it can be said that the proposed approach (i.e. PDE) is expected to outperform the other two.

5 Conclusion

This paper presented a hybrid approach which generates test cases dynamically but in a controlled environment as is the case with symbolic testing. Since the approach applied PSO based search algorithms its effectiveness is much better than that achieved by any algorithm based on random search. We validate our methodology using ten real world test objects. The results were found highly encouraging. We observed that the proposed approach was able to achieve 100% test coverage in nine out of the ten objects. Even in the case of 10^{th} object it was observed that lower performance was not the outcome of any shortcoming on the part of approach but was due to the inherent characteristics of object itself.

References

1. Beizer, B.: Software testing techniques. Dreamtech publication, New Delhi (1990)
2. Bueno, P.M.S., Jino, M.: Identification of potentially infeasible program paths by monitoring the search for test data. In: Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE 2000), France, pp. 11–15 (2000)
3. Coward, P.D.: Symbolic execution and testing. Information and Software Technology 33(1), 53–64 (1991)
4. Deng, X., Robby, Hatcliff, J.: Kiasan/KUnit: Automatic test case generation and analysis feedback for open object-oriented systems. In: Proceedings of the Testing: Academic and Industrial Conference Practice and Research Techniques, Washington, pp. 3–12 (2007)
5. Godefroid, P., Klarlund, N., Sen, K.: DART: Directed Automated Random Testing. In: The Proceedings of Programming Language Design and Implementation (June 2005)
6. Godefroid, P.: Compositional dynamic test generation. In: The Proceedings of ACM Symposium on Principles of Programming Languages (2007)

7. Goldberg, A., Wang, T.C., Zimmerman, D.: Applications of Feasible Path Analysis to Program Testing. In: ISSTA 1994, Seattle, Washington, USA, pp. 80–94 (1994)
8. Khurshid, S., Păsăreanu, C.S., Visser, W.: Generalized symbolic execution for model checking and testing. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 553–568. Springer, Heidelberg (2003)
9. Majumdar, R., Sen, K.: Hybrid concolic testing. In: The proceedings of the 29th International Conference on Software Engineering, Washington DC, pp. 416–426 (2007)
10. McMinn, P.: Search-based Software Test Data Generation: A Survey. Software Testing, Verification and Reliabilit 14(2), 105–156 (2004)
11. Ngo, M.N., Tan, H.B.K.: Detecting Large Number of Infeasible Paths through Recognizing their Patterns. Information and Software Technology 50, 641–655 (2008)
12. Pasareanu, C.S., Mehlitz, P.C., Bushnell, D.H., Gundy-Burlet, K., Lowry, M., Person, S., Pape, M.: Combining unit-level symbolic execution and system level concrete execution for testing NASA software. In: Proceedings of the International Symposium on Software Testing and Analysis, pp. 15–25 (2008)
13. Saxena, P., Poosankam, P., McCamant, S., Song, D.: Loop-Extended Symbolic Execution on Binary Programs. In: ISSTA 2009, Chicago, Illinois, USA, July 19-23 (2009)
14. Sen, K., Marinov, D., Agha, G.: CUTE: a concolic unit testing engine for C. In: Proceedings of Joint Conference of ESEC/FSE 2005, Lisbon, Portugal, pp. 263–272 (2005)
15. Visser, W., Pasareanu, C.S., Khurshid, S.: Test input generation with Java PathFinder. In: Proceedings of the International Symposium on Software Testing and Analysis (2004)
16. Visvanathan, S., Gupta, N.: Generating Test Data for Functions with Pointer Inputs. In: 17th IEEE International Conference on Automated Software Engineering (2002)
17. Watkins, A., Hufnagel, E.M.: Evolutionary test data generation: a comparison of fitness functions. Software Practice & Experience 36, 95–116 (2006)
18. `http://www.softintegration.com`
19. Yuhui, S., Eberhart, R.C.: Parameter Selection in particle swarm optimization. In: The 7th Annual Conference on Evolutionary Programming, San Diego USA

Reactive Power Optimization Based on Particle Swarm Optimization Algorithm in 10kV Distribution Network

Chao Wang[1], Gang Yao[2,*], Xin Wang[1], Yihui Zheng[1], Lidan Zhou[2], Qingshan Xu[3], and Xinyuan Liang[4]

[1] Center of Electrical & Electronic Technology, Shanghai Jiao Tong University, Shanghai, P.R. China, 200240
[2] Department of Electrical Engineering, Shanghai Jiao Tong University, Shanghai, P.R. China, 200240
yaogangth@sjtu.edu.cn
[3] JiLin Electric Power Co. LTD. 130021
[4] Department of Environmental Science, College of Life and Environmental Science, Minzu University of China, Beijing, 100081

Abstract. The optimization of reactive power compensation plays an important role in power system planning and designing. A mathematical model in the 10kV distribution network is established in this paper. Its objective function is the cost of investment in equipment of reactive power compensation and active power loss of the system should be the least. The node voltages beyond limited and the generator reactive power output beyond limited will be expressed in the way of penalty function. In this paper, particle swarm optimization will be used. Using PSO's characteristic of high convergence efficiency, the speed of reactive power optimization will be improved. Using the binary PSO, the algorithm can better adapt to solve the problem.

Keywords: 10kV distribution system, reactive power optimization, PSO.

1 Introduction

Voltage is an important indicator to measure the security and economy of the power system, while the reactive power is an important factor in affecting the voltage level. A reasonable distribution of reactive power is a basic condition to ensure the voltage quality [1]. The rational flow of reactive power is able to maintain reactive power balance, not only ensure the voltage quality, improve the system's security and stability, but also reduce the power loss, access to economic benefits [2].

As an important part of power system's planning, reactive power optimization can utilize voltage to control power system, improve grid stability, reduce network loss and ensure a wider operating margin through reactive power compensation [3,4].

10 kV distribution network has many branches and many customers, distribution transformers have no people on duty. Currently, compensation devices which can switch automatically are few and high cost, most of them have a fixed put-in form and can not be changed when the load changes [5]. In order to achieve the best

* Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 157–164, 2011.
© Springer-Verlag Berlin Heidelberg 2011

compensation effect and not to send reactive power back to the main network, how to choose the node of reactive power compensation and reactive compensation capacity is very important [6].

In order to deal with these problems, there appeared linear programming, nonlinear programming, integer programming, graph theory method, sensitivity analysis, interior point method and network flow method and the simulated annealing algorithm, genetic algorithms, particle swarm optimization which are proposed in recent years [7-9].

Particle swarm optimization is a random optimization algorithm based on swarm intelligence algorithm which is merging in recent years. The algorithm is fast in the initial convergence, but in the latter part, it may be trapped in local optimum. This is the major shortcomings of PSO [10,11].

In this paper, an improved PSO has been presented. The method of dynamic adjustment of the inertia weighted has been added into the traditional PSO algorithm, which the global search capability is far superior to the traditional PSO. Finally, it is applied to reactive power optimization of a 10 kV distribution network system with 17 nodes. And its simulation result shows that this method is effective.

2 The Mathematical Model of Reactive Power Optimization

The purpose of reactive power optimization is to minimize the loss of the entire network and improve voltage quality, to save the cost of system operation, to make the system operated stably and safely. The mathematical model has 3 parts, includes the objective function, the power constraint equations and the variable constraint equations.

2.1 The Objective Function

The objective function of reactive power optimization is varied, including both technical performance indicators and economic indicators. The operation of power system should meet the security and economy. Reactive power optimization regulates generator terminal voltage, adjusts the voltage transformation ratio of voltage transformer and compensation capacitor switching and other control variables in order to take full advantage of the system reactive power, to ensure the voltage quality of the users, to achieve the minimum active power loss of the whole network. The objective function can be expressed as the minimum active power loss:

$$P_{loss} = \sum_{\substack{i\in t \\ j\in N}} G_{ij}(V_i^2 + V_j^2 - 2V_iV_j\cos\delta_{ij}) \tag{1}$$

The state variables (node voltages beyond limited and the generator reactive power output beyond limited) can be expressed in the way of penalty function:

$$\min F = P_{loss} + \sum_{ieco_v} \lambda_{Vj}(\frac{V_j - V_{j\lim}}{V_{j\max} - V_{j\min}})^2 + \sum_{ieco_o} \lambda_{Gi}(\frac{Q_{Gi} - Q_{Gi\lim}}{Q_{Gi\max} - Q_{Gi\min}})^2 \tag{2}$$

where the first section of right-hand side is the active power loss; the second section is to punish the node voltages when beyond limited; the third section is to punish generator reactive power output when beyond limited. λ_{vj} and λ_{ci} are penalty factors of

node voltages other than the node of PT and generator reactive power output when beyond limited. ; co_v is the collection of subscript of node voltage of load which is out of range; co_G is the collection of subscript of generator reactive power output of load which is out of range. $V_{j\lim}$ and $Q_{Gi\lim}$ can be expressed as:

$$V_{j\lim} = \begin{cases} V_{j\max} & V_j > V_{j\max} \\ V_{j\min} & V_j < V_{j\min} \\ V_j & V_{j\min} < V_j < V_{j\max} \end{cases} \tag{3}$$

$$Q_{Gi\lim} = \begin{cases} Q_{Gi\max} & Q_{Gi} > Q_{Gi\max} \\ Q_{Gi\min} & Q_{Gi} < Q_{Gi\min} \\ Q_{Gi} & Q_{Gi\min} < Q_{Gi} < Q_{Gi\max} \end{cases} \tag{4}$$

2.2 The Power Constraint Equations

Node power balance equation is the equality constraint:

$$P_i = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos \delta_{ij} + B_{ij} \sin \delta_{ij}) \tag{5}$$

$$Q_i = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos \delta_{ij} - B_{ij} \sin \delta_{ij}) \tag{6}$$

where P_i, Q_i are the active and reactive power which is injected by node i; V_i, V_j are the voltage amplitudes of node i and node j; G_{ij}, B_{ij}, δ_{ij} are conductance, susceptance, angle difference between voltage phase between node i and node j; N indicates the nodes which are connected with node i directly.

2.3 The Variable Constraint Equations

Variable constraints are inequality constraints:

$$\begin{cases} V_{Gi\min} \le V_{Gi} \le V_{Gi\max} \\ Q_{Gi\min} \le Q_{Gi} \le Q_{Gi\max} \\ T_{k\min} \le T_k \le T_{k\max} \\ Q_{Ci\min} \le Q_{Ci} \le Q_{Ci\max} \\ V_{i\min} \le V_i \le V_{i\max} \end{cases} \tag{7}$$

where V_{Gi} is the generator terminal voltage; $V_{Gi\min}$,$V_{Gi\max}$ are the upon limit and lower limit of the generator terminal voltage; Q_{Gi} is the generator reactive power output; $Q_{Gi\min}, Q_{Gi\max}$ are the upon limit and lower limit of the generator reactive power output; T_k is the position of the adjustable transformer tap; $T_{k\min}$,$T_{k\max}$ are the upon limit and

lower limit of the adjust ratio of the transformer; Q_{Ci} is the capacitive reactive compensation capacity; $Q_{Ci\min}, Q_{Ci\max}$ are the upon limit and lower limit of the compensation capacity in node i; V_i is the voltage of node i; $V_{i\min}, V_{i\max}$ are the upon limit and lower limit of the voltage amplitude of node i.

3 Particle Swarm Optimization

PSO (Particle Swarm Optimization) is a new evolutionary algorithm which is developed in recent years. It was originally developed by Dr. Kennedy and Dr. Eberhart in 1995 which is inspired by the research of artificial life. It is an evolutionary computation technique based on swarm intelligence which is proposed when simulating birds' behavior of migration and the cluster during foraging.

In PSO, the solution of each optimization problem is a bird in the search space. We call it "particle". Each particle has a fitness value which is determined by the function to be optimized and a speed to determine their flying direction and distance. Then particles will search in the solution space search, following the current best particle. However, due to the defects in the basic PSO (as in theory, absolute convergence has not be proved, it has the danger of falling into local optimal) and actual use (such as requirements of the computing speed) and other reasons, we have to do some improvements to the basic PSO, so that PSO can be adapted to the reactive power optimization.

3.1 The Introduction of Inertia Factor

In PSO, each particle is the optimization solution in the feasible region. First, it initializes a group of particles randomly. Then, these particles adjust their direction and location to search for the optimal solution by learning two "extreme" in the iteration process. These two "extreme", one is individual extreme value, which is its own optimal value of the current; the other is the global extreme value, the optimal solution of all particles currently. In the searching process, the particles adjust their speed V_{id} and location X_{id} according to these two values.

To speed up the convergence rate of PSO, with the inertia factor, the formula of particles' velocity and displacement is:

$$V_{id}^{k+1} = \omega V_{id}^{k} + C_1 rand()(pBest_{id}^{k} - X_{id}^{k}) + C_2 rand()(gBest_{id}^{k} - X_{id}^{k}) \tag{8}$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \tag{9}$$

where ω is called the inertia factor; X_{id} is the particle's position; V_{id} means the particle's speed; C_1, C_2 are acceleration factors; $rand()$ is a random number between 0 and 1; $pBest_{id}$、$gBest_{id}$ are the best particle position and the best position of the global.

ω is a scale factor which is related with the previous speed, it controls how the next iteration's velocity is impacted of the previous iteration's speed value. A larger ω can strengthen the global search capability in PSO; a smaller ω can enhance the local search capability. So if using a same value of ω in the whole process of PSO iterations, the algorithm can not be easily suitable for global search and local search. In this paper, ω will decrease linearly from 0.9 to 0.4 in the whole iteration process:

$$\omega = 0.9 - \frac{0.9 - 0.4}{N_{\max}} \times n \tag{10}$$

where $N_{\max}$ is the total number of iterations; n is the current number of iterations.

3.2 The Construction of Particles in Reactive Power Optimization

In PSO, each particle is instead of each solution in the reactive power optimization. Using the objective function, the fitness of each particle can be determined: the higher fitness, the more superiority of the particles. Each particle searches for the optimal solution according to its own and other particles' flying experience in the solution space.

When using PSO to determine compensation point, each node only has two states: a compensation point or not. So binary coding is suitable for PSO: each particle represents a combination of compensation state. Using binary coding, 0 means this node has no compensation, the node 1 means that it has compensation.

3.3 Initialization of PSO

For the binary coded PSO, the initial of the particles' position is randomly selected certain nodes as the initial compensation nodes. Because of the actual situation of the distribution network, the compensation nodes should not be too many; otherwise the economy will be poor. According to the experience, 3-5 nodes will be appropriate.

The velocity of particles in the distribution network reconfiguration problem has no actual physical meaning. It represents the different between the current states of the switching portfolio with the best combination states. The speed will be faster if the difference is greater. In the binary coded PSO, the particle velocity is on behalf of a dimensional coordinates of the probability of this state or another state. In this paper, Sigmoid function will be used to update the particles' velocity:

$$Sigmoid(V_{id}) = \frac{1}{1 + e^{-V_{id}}} \tag{11}$$

4 System Examples

Take a distribution network of 17 nodes and assume the voltage of the first outlet end of the root node is a constant value 10.5kV.

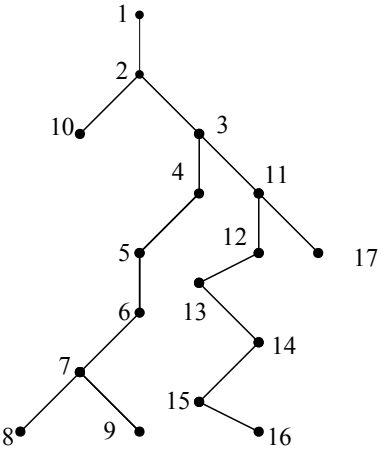


Fig. 1. A 17-node System

All raw data of the system is as follow:

Table 1. Raw Data

Start Node	Last node	Resistance(Ω)	Reactance(Ω)
1	2	0.033117	0.035511
2	3	0.026563	0.028484
2	10	0.028350	0.023184
3	4	0.015593	0.012751
3	11	0.008366	0.008971
4	5	0.014459	0.011824
5	6	0.021263	0.017388
6	7	0.014459	0.011824
7	8	0.021263	0.017388
7	9	0.017483	0.014297
11	12	0.012062	0.012933
11	17	0.023814	0.010293
12	13	0.011852	0.012709
13	14	0.011155	0.011962
14	15	0.011504	0.012335
15	16	0.009691	0.010392

Table 2. Node voltage before and after the compensation

node	before	after
1	10.500	10.500
2	10.324	10.329
3	10.220	10.228
4	10.214	10.218
5	10.210	10.211
6	10.199	10.208
7	10.192	10.207
8	10.184	10.199
9	10.190	10.212
10	10.301	10.306
11	10.188	10.202
12	10.157	10.175
13	10.127	10.148
14	10.100	10.125
15	10.083	10.112
16	10.082	10.110
17	10.164	10.180

Table 2 shows the voltage of each node has been improved in varying degrees after the compensation. In which the lower voltage before compensation will be improved the larger, such as node 15, 16; the higher voltage before compensation will be improved the smaller, such as Node 2.

Table 3. Reactive power compensation added

Node	2	3	7	11
Capacity added (*kVar*)	5.25	3.93	19.98	8.40

Table 3 shows that the reactive power compensation capacity added of each node by the end is: node 2, 5.25 kVar; node 3, 3.93 kVar; node 7, 19.98 kVar; node 11, 8.40 kVar.

The above analysis shows that after adding reactive power compensation, the system voltage level has been markedly improved. Therefore, the calculated compensation plan is reasonable.

Through comparative analysis of before and after the compensation, we can see voltage level and the net loss of the grid has been greatly improved, the equipment investing cost of the reactive power compensation has been saved, while the reactive power has been well improved. The reactive power optimization can reduce the whole

Net Loss, improve the system voltage level, and thereby raise the level of economic, security of grid operation.

5 Conclusion

In this paper, mathematical model in the 10kV distribution network has been established. Its objective function is the cost of investment in equipment of reactive power compensation and active power loss of the system should be the least. The node voltages beyond limited and the generator reactive power output beyond limited can be expressed in the way of penalty function. The convergence speed and convergence precision has increased effectively, the system voltage security been improved by using improved PSO to select the optimal compensation node. The improved PSO makes PSO more suitable for reactive power optimization in the distribution network, has theoretical and practical significance.

References

1. MakSioe, T.: Propagation of Transientsina Distribution Network. IEEE Transaction on Power Delivery 8(1), 337–343 (1993)
2. Mantawy, A.H., AI-Ghamdi, M.S.: A New ReactivePower Optimization Algorithm. In: 2003 IEEE Bologna PowerTech Conference, Bologna, Italy (2003)
3. Durairaj, S., Fox, B.: Evolutionary computation based reactive power optimization. In: IET-UK International Conference on Information and Communication Technology in Electrical Sciences (ICTES 2007), pp. 120–125. Dr.M.G.R.University, Chennai (2007)
4. Yoshida, H., Kawata, K., Fukuyama, Y., et al.: A particle swarmoptimization for reactive power and voltage control securityassessment. IEEE Trans. on Power Systems 15(4), 1232–1239 (2000)
5. Yu, X., Li, Y., Xiong, X.: Optimal shunt capacitorplacement using particle swarm optimization algorithm withharmonic distorts ion consideration. Proceedings of the CSEE 23(2), 26–30 (2003)
6. Zhou, S., Zhu, L., Guo-jiu, et al.: The power systemvoltage stability and control. China ElectricPower Press, Beijing (2004)
7. Guo, Y., Wen, J.: A new power system reactive poweroptimization algorithm. Electric Power Survey and Design 12(1), 66–70 (2005)
8. Tan, T.-l., Zhang, Y.: Reactive power optimizationbased on genetic/ tabu search hybrid algorithm. PowerSystem Technology 28(11), 57–61 (2003)
9. Eberhart, R.C., Shi, Y.: Comparing inertia weights and const riction factors in particle swarm optimization. In: Proc.of the IEEE Conf. on Evolutionary Computation, pp. 84–88. IEEE Service Center, California (2000)
10. Naka, S., Genji, T.: A hybrid particle swarm optimization for distribution state estimation. IEEE Trans. on Power Systems 18, 60–68 (2003)
11. Zhang, W.-J., Xie, X.-F.: DEPSO Hybrid Particle swarms with differential evolution operator. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 4, pp. 3816–3821 (2003)

Clustering-Based Particle Swarm Optimization for Electrical Impedance Imaging

Gang Hu, Min-you Chen, Wei He, and Jin-qian Zhai

State Key Laboratory of Power Transmission Equipment & System Security and New Technology, Chongqing University, Chongqing 400044, China

Abstract. An attempt has been made in this paper to solve the non-linear and ill-posed Electrical Impedance Tomography (EIT) inverse problem using clustering-based particle swam optimization (PSO). To enhance optimal search capability in such an ultra high dimension space and improve the quality of the reconstructed image, an adaptive PSO algorithm combined with a modified Newton–Raphson algorithm and a conductivity-based clustering algorithm was proposed. The modifications are performed on the reduction of dimension by dividing all mesh into clusters and initializing particles using the result of the modified Newton–Raphson type algorithm. Numerical simulation results indicated that the proposed method has a faster convergence to optimal solution and higher spatial resolution on a reconstructed image.

Keywords: Particle Swarm Optimization, Clustering, Reconstruction Algorithm, Electrical Impedance Tomography.

1 Introduction

The electrical impedance tomography (EIT) problem is an inverse problem that the internal conductivity of an inaccessible region can be estimated through currents injected into the region and the corresponding voltages measured on the surface. EIT is very suitable for early detection, diagnosis, prediction and evaluation after healing for nerve diseases and malignant tumors [1]. Essentially, the EIT problem is a non-linear inverse problem and is severely ill-posed. The use of a Newton–Raphson type solution is widespread in EIT, the Newton's One-Step Error Reconstructor (NOSER) algorithm being a well known example [2]. However, the linearized solution will only be accurate when the true conductivity is close to the initial estimate and the process is likely to be trapped in a local minimum.

Recently, some global optimizing evolutionary algorithms, such as GA [3], [4] and DE [5], were put forward to solve the inverse problem. Although these algorithms are relatively expensive in terms of computing time and resources, the spatial resolution of the reconstructed images could be improved. In the years since the introduction of PSO as a new method for global optimization [6], the PSO algorithm has been found to be successful in a wide variety of optimization tasks, including neural network training [7] and function minimization [8]. In this paper, a clustering-based PSO reconstruction method for EIT was proposed, which used the result of NOSER as prior knowledge to

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 165–171, 2011.
© Springer-Verlag Berlin Heidelberg 2011

initialize the population and reduce the dimension of problem space by clustering the meshes with a similarity factor. The convergence of the PSO was accelerated and the quality of imaging was improved.

2 The EIT Problem

The principle of operation of EIT is shown in Fig. 1, where low frequency current is injected through a pair of electrodes and voltages are captured on others. After sixteen injecting and measuring, the conductivity distribution would be evaluated. The potential distribution function φ and conductivity distribution function σ in the region Ω are governed by the Laplace equation (1) subject to the boundary condition (2).

$$\nabla[\sigma(x,y)\cdot\nabla\varphi(x,y)]=0 \quad (x,y)\in\Omega \tag{1}$$

$$\sigma(x,y)\frac{\partial\varphi(x,y)}{\partial n}=j(x,y) \quad (x,y)\in\partial\Omega \tag{2}$$

where j is the current density applied on the boundary, ∂Ω is the boundaries of Ω and n is the unit outward normal vector to the boundary surface.

This inverse problem of EIT also can be presented as minimizing the objective function E(σ) with respect to σ,

$$E(\sigma)=\sum_{i=1}^{N}\sum_{j=1}^{N-3}(V_{ij}-U_{ij}(\sigma))^2\ , \tag{3}$$

where V_{ij} are the measured boundary voltages and U_{ij} are the calculated boundary voltages. This suggest the viability of employing PSO for the solution of the EIT problem.

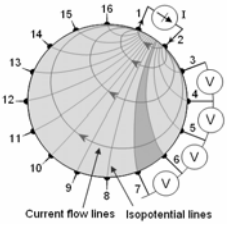


Fig. 1. Principle of Operation of EIT

3 Clustering-Based PSO for Image Reconstruction in EIT

There are many modified versions of PSO by improving convergence property to a certain problem. While, the modified PSO algorithm suggested in [9] was used to solve

the EIT inverse problem in this paper. In the simple implementation of PSO in EIT, a swarm (particles) of tentative solutions (EIT images) is generated, usually at random. Each particle's position vector consists in m-tuple of conductivity values (m is the number of elements discretizing the imaging region), i.e., the EIT particle's position vector is a sequence of m conductivities. A fitness value is computed for each individual. In the present case, the fitness function is equation (3). In EIT imaging, it normally requires a large number of elements in the FE model (e.g. m=864 in our FE model shown in Fig.2) to obtain satisfactory resolution in the imaging region. In such a high-dimensional space, the PSO algorithm is often failed in searching the global optimal solution.

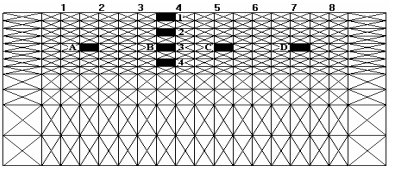


Fig. 2. The FE model used in computer simulation

In fact, the existence of biological tissue is always classified and similar tissues have similar electrical properties. This enlighten us that if all the elements can be reasonably assigned into several subsets under certain similarity measures without losing key information, it will reduce the dimension of the problem space significantly and improve the convergence of the PSO algorithm. Furthermore, after the one-step iteration of the NOSER algorithm, absolute conductivity values of meshes cannot be determined but some useful information on the objective images can be obtained which can be used as prior information of the optimization problem.

To increase the feasibility of the pre-clustering in EIT reconstruction, a similarity factor was proposed to control the clustering, which includes two distance measures, the one is concerned with the conductivity value and the other is concerned with the adjacency of space location of elements. These two distance measures represented by VD and SD were defined as follows:

$$VD = \left|\sigma_{ei} - \sigma_{cj}\right|;\ SD = \sqrt{(x_{ei} - x_{cj})^2 + (y_{ei} - y_{cj})^2}\ , \tag{4}$$

where σ_{ei} is the conductivity of the i^{th} element, σ_{cj} stands for the conductivity of the j^{th} cluster, x_{ei} and y_{ei} are the coordinates of the i^{th} element, x_{cj} and y_{cj} stand for the coordinates of the j^{th} cluster. The conductivity of a cluster is defined as follows:

$$\sigma_{cj} = \sum \sigma_i / G_j\ , \tag{5}$$

where σ_i is the conductivity value of an element belonging to cluster *j*, and G_j is the number of elements in cluster *j*. The coordinates of a cluster are defined as follows:

$$x_{cj} = \sum x_i / G_j\ ;\ y_{cj} = \sum y_i / G_j\ , \tag{6}$$

where x_i and y_i are the coordinates of an element belonging to cluster j and G_j is the number of elements in cluster j.

Using the distance measures VD and SD, a self-organizing network based clustering algorithm [10] is used to establish the clusters. With different thresholds for VD and SD, different number of clusters will be divided. By the pre-clustering, the dimension of the PSO algorithm's searching space was greatly reduced from m to N, where N is the number of clusters which is much smaller than the number of elements m. Thus, the position vector of i^{th} individual in the population was represented as x_i=[x_{i1},…, x_{ij},…, x_{iN}]=[σ_{i1},…, σ_{ij},…, σ_{iN}], where σ_{ij} denotes the conductivity of j^{th} cluster. Instead of initializing all particle's position vector randomly, one individual's position vector was initialized by the conductivity distribution [σ_1,…, σ_j,…, σ_N], which was calculated through equation (5) with the result of the NOSER algorithm and self-organizing network based clustering algorithm.

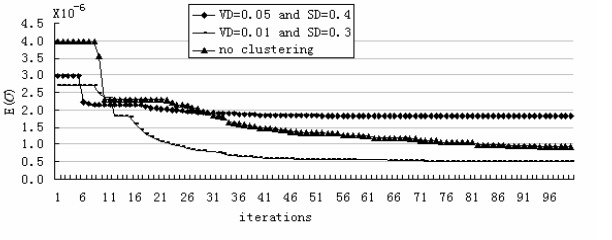


Fig. 3. The convergent performance of different threshold for VD and SD

To preliminarily estimate the effect of the pre-clustering on the convergence of the PSO, a simulation experiment for EIT image reconstruction was conducted. Eight evenly spaced electrodes array put on the top boundary of a rectangle region and the 864 elements, 463 nodes FE Model was used to discretize it. The conductivity of the black region 3 or B shown in Fig.2 was set to 5 S/m and the others were set to 1 S/m. Two different thresholds of VD and SD were taken into consideration and these two cases were compared with the non-clustering case. The convergence performance of the PSO algorithm with different clustering results was shown in Fig.3. It can be seen that the PSO with pre-clustering process (the lines with dot markers and diamond markers) converged faster than the PSO algorithm without clustering (the line with triangle markers). But without sufficient clusters, it is easy for the algorithm to be premature (see the line with diamond markers for iterations > 6). So the values for VD and SD should to be optimized for different imaging situation.

In this paper, the actual implementation in EIT (clustering based PSO in the following) consists of a two-stage PSO. In the first stage, particles not only search the optimal conductivity values but also the optimal VD and SD values. So individual's position vector includes N conductivity values, the VD value and the SD value, i.e., x_i=[x_{i1}, x_{i2}, …, x_{iN}, $x_{i(N+1)}$, $x_{i(N+2)}$] =[σ_{i1}, σ_{i2},…, σ_{iN}, VD_i, SD_i]. For different VD and SD values the N is different, so the dimension of the position vector is different for each particle and even for each iteration of the same particle. In this stage, the two important values, individual-best and global-best for conductivity-position need a special process.

In this paper, at every iteration, the individual-best and global-best are first mapped into non-clustered space and then set these two values of each new cluster by the average values of all the elements belonging to the cluster. The two best values of a cluster are defined as follows:

$$p_{bestj} = \sum p_{besti} / G_j \, ; g_{bestj} = \sum g_{besti} / G_j \, , \tag{7}$$

where p_{besti} is the individual-best value of an element belonging to cluster j, g_{besti} is the global-best value of an element belonging to cluster j, and G_j is the number of elements belonging to cluster j.

The second PSO finally solves the EIT problem by searching for the conductivity distribution without clustering. In this stage, one population was initialized by the conductivities computed in the first stage and others generated randomly.

The first stage PSO procedure can be described as following steps:

Step1. Initialisation

a) Set population number and iteration number.
b) Randomly set the threshold of VD and SD for each particle.
c) Use a self-organizing network to generate element clusters using the given VD, SD values based on the conductivity values calculated by NOSER.
d) Initialize the first particle position using the conductivity values calculated through the NOSER algorithm and the others position and velocity of the particles within the pre-defined decision variable range. Set the maximum allowable velocity. Set individual-best and global-best position.

Step2. Evaluation

Evaluate each particle in the current population using Pareto-based fitness assignment strategy. Update individual-best and global-best.

Step3. New particles generation

a) Calculate individual-best and global-best according to clusters.
b) Calculate the new velocity and new position.
c) According to new VD and SD value re-cluster elements to clusters.
d) Calculate the objective function values for all the new particles. Combine all position and new position (2N particles) together and store them in a temporary list.

Step4. Non-dominated Sorting.

Step5. Select particles for next iteration.

Step6. If satisfied the control condition, execute mutation operation, otherwise go to Step7.

Step7. If satisfied the control condition, go to Step2.

Step8. Store the non-dominated solutions from the final population.

The proposed method was tested with simulation experiment shown in Fig.2. The background conductivity was set to 1 S/m and the target conductivity was chosen to 5 S/m. To study the spatial resolution in the X direction parallel to the electrode arrays, four locations A~D shown in Fig.2 were tested. To study the spatial resolution in the Y direction, the depths of positions 1 to 4 shown in Fig.2 were tested.

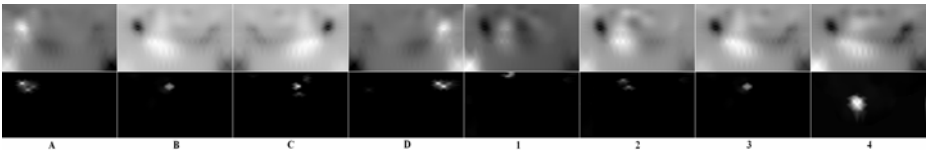


Fig. 4. The reconstruction images using NOSER and clustering-based PSO

The images reconstructed by NOSER (the upper images) and clustering-based PSO (the bottom images) were shown in Fig.4. In these figures, the conductivity value was represented by brightness, i.e. the objective with higher conductivity corresponds to brighter region. Fig.4 shows that the clustering-based PSO can locate the target more accurately and reconstruct images more clear than NOSER.

For each location, the image reconstruction was run 10 times independently with random initial conditions and adding randomly distributed noise (the maximum signal-to-noise ratio was 30DB) to the calculated voltages. It is clearly shown that the clustering-based PSO can improve the reconstruction image on the accuracy of solution remarkably.ï

4 Conclusions and Further Developments

PSO appear to be a promising tool for the solution of the EIT problem. Although PSO is presently unsuitable for real-time tomographic applications, the exploitation of prior knowledge has the potential to produce better reconstructions. The proposed clustering-based PSO-EIT method is able to finding optimal solution of reconstruction that can improve imaging resolution.

Further research should be done to improve the stability of the proposed method. The successful cases should be studied to obtain some experiential knowledge to optimize VD and SD.

Acknowledgements

The authors gratefully acknowledge the Project No.CDJXS11151154 supported by the Fundamental Research Funds for the Central Universities.

References

1. Brown, B.H.: Electrical impedance tomography. Journal of Medical Engineering & Technology 27(3), 97–108 (2003)
2. Lionheart, W.R.B.: EIT reconstruction algorithms: pitfalls, challenges and recent developments. Physiol. Meas. 25, 125–142 (2004)
3. Kimt, H.C., Mood, D.C., Kimtt, M.C., Kimttt, S., Leettt, Y.J.: Improvement in EIT Image Reconstruction using Genetic Algorithm. In: Proceedings of the American Control Conference, Anchorage, pp. 3858–3863 (2002)
4. Olmi, R., Bini, M., Priori, S.: A Genetic Algorithm Approach to Image Reconstruction in Electrical Impedance Tomography. IEEE Transactions on Evolutionary Computation 4(1), 83–88 (2000)
5. Li, Y., Rao, L., He, R., Xu, G., Wu, Q., Yan, W., Dong, G., Yang, Q.: A Novel Combination Method of Electrical Impedance Tomography Inverse Problem for Brain Imaging. IEEE Transactions on Magnetics 41(5), 1848–1851 (1848)
6. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. Neural Networks. In: Proc. IEEE Inter. Conf. on Neural Networks, Perth, pp. 1942–1948 (1995)
7. Engelbrecht, A.P., Ismail, A.: Training product unit neural networks. Stability Control: Theory Appl. 2(1-2), 59–74 (1999)
8. Parsopoulos, K.E., Vrahatis, M.N.: On the Computation of All Global Minimizers Through Particle Swarm Optimization. IEEE Trans. Evol. Comput. 8(3), 211–224 (2004)
9. Chen, M., Wu, C., Fleming, P.J.: An evolutionary particle swarm algorithm for multi-objective optimization. In: Processing of the 7th World Congress on Intelligent Control and Automation, pp. 3269–3274. IEEE Press, Los Alamitos (2008)
10. Linkens, D.A., Chen, M.: Hierarchical Fuzzy Clustering Based on Self-organising Networks. In: Proceedings of World Congress on Computational Intelligence (WCCI 1998), vol. 2, pp. 1406–1410. IEEE, Piscataway (1998)

A PSO- Based Robust Optimization Approach for Supply Chain Collaboration with Demand Uncertain

Yutian Jia, Xingquan Zuo, and Jianping Wu

Computer School, Beijing University of Posts and Telecommunications, Beijing, China
xiaojia164@sina.com, zuoxq@bupt.edu.cn,
jianpingwu.soton@gmail.com

Abstract. A robust optimization approach is proposed to solve the problem of supply chain collaboration under a demand uncertain environment. The proposed approach is universal and able to adapt to various demand models. First, the uncertain demand is described by a set of sampling sequences, and the total cost of supply chain is calculated based on these sequences to evaluate a collaboration scheme. Then a particle swarm optimization (PSO) is employed to find the optimal collaboration scheme which leads to a minimum total cost of supply chain. Numerical experiments show that the proposed approach can produce a robust solution that is insensible to the effect of demand uncertainty.

Keywords: Supply chain collaboration, Robust optimization, Particle swarm optimization.

1 Introduction

In recent years, inter-enterprise collaborative decision-making has become an important research direction in the field of supply chain management. The operational efficiency of supply chain is improved and its costs are reduced through the collaboration decision between members in supply chain. Some researchers show great interest in supply chain collaboration. Fung et al[1], Li et al[2] made a review on the mechanism of supply chain coordination respectively. Barbarosoglu et al [3], Ozdamar et al[4] considered the large-scaled collaborative problem, solved those complicated problems effectively and got a near-optimal solution.

Most of the literatures consider minimizing the total cost of the supply chain as the collaborative decision-making target. However, calculating the total cost of supply chain is difficult due to the demand uncertainty, and sometimes the total cost function is hard to express. Researchers have made various assumptions and limitations to deal with the demand uncertainty. Some suppose the demand is determined, so the calculation of total costs can be simplified. Darwish et al[5] studied the case of one supplier and multiply retailers, each retailer's demand rate is supposed to be determined and distribution from the supplier to all the retailers must occur at the same time. The demand was also assumed to be determined in [6] and the inventory and distribution decision was made based on this assumption. In addition, some assume that the demand follows certain probability distribution. Kang et al[7] studied the collaborative problem

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 172–181, 2011.
© Springer-Verlag Berlin Heidelberg 2011

of one supplier and one retailer. The number of customers arriving at the retailer (in a unit time) follows a Poisson distribution with a known mean value, and the mean and variance of demand quantities of customers are also known. The goal is to minimize the expected long-run average cost. Amanda et al[8] studied the case that the demand of retailer is stochastic and gave a closed-form approximate solution.

For a supply chain with uncertain demand, only a suboptimal collaborative decision can be obtained if the demand is assumed to be determined. Assuming the demand follows a probability distribution will reduce the universality of total cost calculation. Moreover, the probability distribution of uncertain demand is actually difficult to estimate.

In this paper, a PSO based robust optimization approach is proposed to solve the problem of supply chain collaboration. The approach can produce a collaboration decision scheme that is insensible to demand uncertain. First, some demand instances are generated randomly, and the total cost of supply chain are calculated by these instance to evaluate a collaboration decision scheme. Then a PSO is used to optimize the decision schemes to obtain an optimal robust decision scheme. Experimental results show that the proposed approach can obtain a more stable decision scheme, which is robust to the demand uncertainty of supply chain.

2 Problem Description

We consider a supply chain that contains a supplier and a retailer, and the customer demand is uncertain, as shown in Fig. 1. The retailer holds an inventory to satisfy the demand of the customer, and unsatisfied demand will result in an unfulfilled penalty cost. The retailer has a reorder point r and an order quantity q. When the inventory level is lower than r, the retailer orders quantity q goods from the supplier, the order takes a lead time to arrive. The supplier also has a reorder point R and an order quantity Q. When the inventory level is lower than R, the supplier orders quantity Q goods from the upstream supplier, the order also takes a lead time to arrive. Fig. 2 shows inventory level changes for the retailer and the supplier. The retailer and the supplier must disburse the inventory holding cost, the order cost and the unfulfilled penalty cost, respectively. The total cost of the supply chain is sum of the cost of retailer and supplier. The collaboration decision purpose is to minimize the total cost of supply chain.

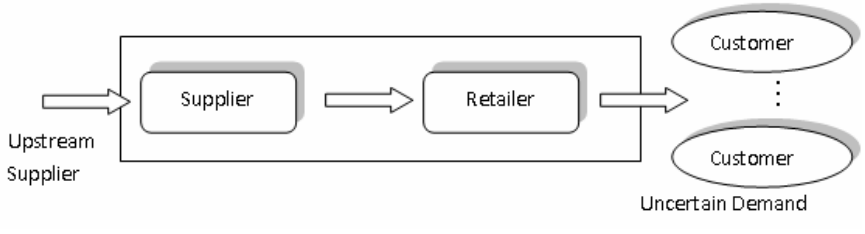


Fig. 1. Structure of the supply chain

For the retailer, the part which customer's demand cannot be satisfied is considered to be the unfulfilled quantity; but for the supplier, as long as the inventory level is lower than the retailer's order quantity, it will be out of stock. The supplier does not deliver goods to the retailer, and the unfulfilled quantity is always the retailer's order quantity. This is because supplier's fixed scale batches of distributions may save the cost. Hence the supplier's best order quantity should be an integral multiple of the retail's order quantity. It is easy to explain with the reduction to absurdity: if the supplier's order quantity Q is not the integral multiple of q, then let $Q = nq + k$ ($0<k<q$). It is obviously to know $Q = nq$ has the same supply capacity, but $Q = nq$ takes up fewer inventories. Since supplier's inventory is reduced by q units each time, therefore the best reorder point R is also an integral multiple of q. Therefore, supplier's actual decision variables are not Q, R, but are the multiple of q, denoted by n_q and n_r, respectively.

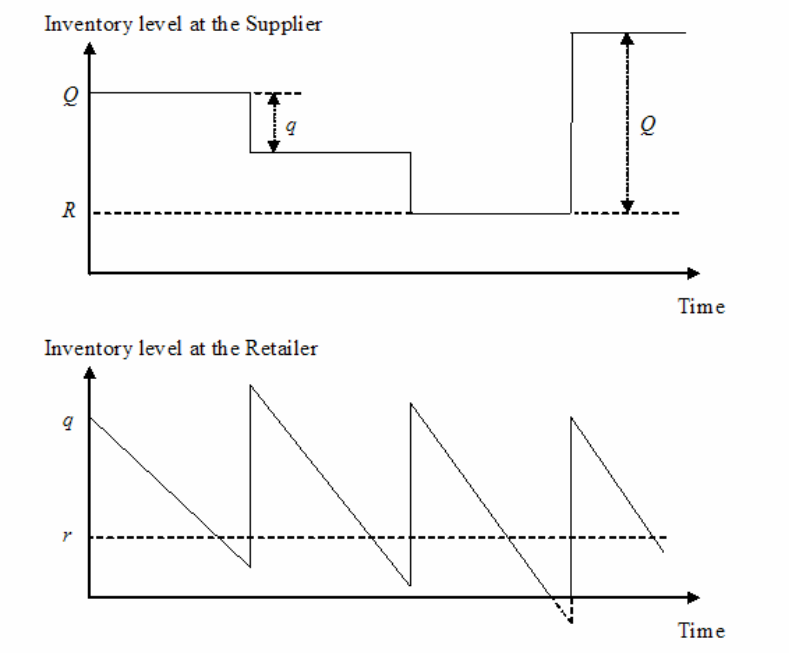


Fig. 2. Schematic of inventory level

3 The Cost Calculation

As mentioned above, $\{n_q, n_r, q, r\}$ is the set of decision variables. Since the demand is uncertain, it is difficult to calculate the total cost directly. In this paper, a set of demand sampling sequences is produced randomly to reflect the demand uncertainty, and used to calculate the total cost of supply chain to evaluate a set of decision variables.

3.1 Calculating the Total Cost

The notations used in the cost calculations are as follows.

T	length of time horizon (days)
N_{ord}	number of order during the T days
N_{unf}	number of out-of-stock times during the T days
I_k	retailer's inventory level in the kth day (k = 1, 2, …, T)
U_k	retailer's kth unfulfilled quantity (k = 1, 2, …, N_{unf})
D_k	demand of the customers in the kth day(k = 1, 2, …, T)
hOf	unit order fixed cost of the retailer
hOv	unit order variable cost of the retailer
hU	unit unfulfilled penalty cost of the retailer
hI	unit inventory holding cost of the retailer per day
LR	lead time of the retailer

The total cost of the retailer is calculated as follows.

Step 1 Input *T, hOf, hOv*, *hU*, *hI and LR*.

Step 2 Produce a sample sequence (D_1, D_2, ..., D_T) randomly according to some probability distribution. Each element in the sequence represents the demand for a day.

Step 3 Let the initial inventory level of the retailer to be q.

Step 4 Inventory level of the retailer is decreased by D_k (k = 1, 2, …, T) everyday. When the inventory level is lower than r, an order is placed by the retailer and the goods will arrive in a lead time LR. If the on-hand inventory of the retailer is not sufficient to satisfy customers' demands, the unfulfilled part is the quantity of shortage.

Step 5 The inventory holding cost for one day is calculated by multiply the unit inventory cost and inventory level of the day. Hence, in the kth day, the inventory holding cost C_I_k is

$$C_I_k = I_k * hI \tag{1}$$

Hence the total inventory holding cost C_I is

$$C_I = \sum_{k=1}^{T} C_I_k = hI \sum_{k=1}^{T} I_k \tag{2}$$

Step 6 The order cost is composed of two parts: the fixed part of each order and the variant part which changes linearly with the order quantity. Hence the total order cost C_O is

$$C_O = N_{ord}(hOf + q * hOv) \tag{3}$$

Here, $hOf + q*hOv$ is the order cost each time.

Step 7 It is difficult to estimate the loss when inventory is out of stock, so we use an unfulfilled penalty cost here, which is proportional with the unfulfilled quantity. If the kth shortage occurs in the mth day then

$$U_k = D_m - I_m \tag{4}$$

Hence the total unfulfilled penalty cost C_U is

$$C_U = hU \sum_{k=1}^{N_{unf}} U_k \tag{5}$$

Step 8 Denote the total cost of the retailer is TC_R

$$TC_R = C_I + C_O + C_U = hI \sum_{k=1}^{T} I_k + N_{ord}(hOf + q^* hOv) + hU \sum_{k=1}^{N_{unf}} U_k \tag{6}$$

The pseudo codes for this approach is given in Fig. 3

```
Procedure: Calculate total cost for the retailer
    Begin
        Input T, hOf, hOv, hU, hI and LR.
        Generate a sample sequence (D1, D2, ..., DT);
        //set initial inventory
        I1 = q;
        C_I= C_R = 0;
        k = 1;
        While (k <= T) do
            //check the goods in transit, if any
            If (goods sent from the supplier arrive) then
                Ik = Ik + q;
            End if
            // satisfy customers' demands
            If (Ik >= Dk) then
                Ik = Ik - Dk ;
            Else        //out of stock
                UR = Dk - IRk;
                CUR = CUR + hUR*UR;
                Ik = 0;
            End if
            //decide whether to order
            If (Ik <= r) then
                Place an order to the supplier;
                Nord = Nord + 1;
            End if
            // accumulate the inventory cost
            C_I = C_I+ hI*Ik;
            k = k + 1;
        End Loop
        //count all cost
        C_O= Nord(hOf + q*hOv);
        TCR = C_O+ C_I+ C_U;
    Stop
```

Fig. 3. Calculation method for the total cost of the retailer

The cost calculation method of the supplier is similar to that of the retailer and the total cost of the supplier denoted as TC_S is also obtained. Hence the total cost of supply chain TC is

$$TC = TC_R + TC_S \tag{7}$$

3.2 Evaluation of the Decision Variables

In order to evaluate the decision variables, we generate N sampling sequences randomly which follow some probability distribution. Each sampling sequence will be used to obtain a total cost denoted as TC_k (k = 1, 2, …, N) according to the approach given in part 1.1.

The average value of TC_k denoted by $\overline{TC}$ is

$$\overline{TC} = \frac{1}{N}\sum_{k=1}^{N} TC_k$$

The flow chart is given in Fig. 4.

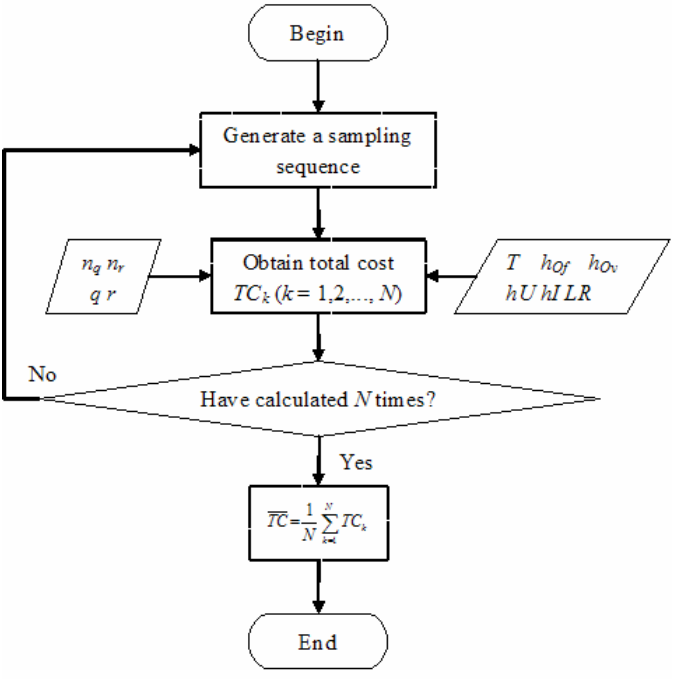


Fig. 4. The flow chart of the evaluation method

4 Optimize the Decision Variables by the PSO

4.1 Description of the Standard PSO

For a standard PSO, there are M particles in n-dimensional space and the position of each particle represents a potential solution. The following notations are used in the PSO:

$X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$	current position of the ith particle
$V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$	current speed of the ith particle
$P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$	the best position that the ith particle passed
$P_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$	the best position that all the particles passed

In the tth generation, the evolution equation for the jth dimension of the ith Particle is

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 \text{rand}()(p_{ij}(t) - x_{ij}(t)) + c_2 \text{rand}()(p_g(t) - x_{ij}(t)) \tag{8}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{9}$$

Here, ω is the inertia weight, $c_1$and c_2 are the acceleration constants. The rand() is a function used to generate random numbers in the range of [0, 1].

4.2 Algorithm Steps

In this paper, a decision variable set $\{n_q, n_r, q, r\}$ is represented by a 4-dimensional particle in the PSO. The steps of the PSO are as follows.

Step 1 Initialize position and speed. Generate M 4-dimensional particles, whose initial position and speed are produced randomly.

Step 2 Calculate the fitness value. For each particle, calculate its total cost by the approach proposed in part 3. Fitness of the particle is the reciprocal of the total cost.

Step 3 For each particle, compare its current position and its historical best position, if the former is better, then let the historical best position equal to be the current position.

Step 4 For each particle, compare its current position and the global best position, if the former is better, then let the global best position equal to be the current position.

Step 5 Update position and speed of each particle according to the evolution equations (8) and (9).

Step 6 If the termination condition is not satisfied, then return to Step 2; otherwise, the algorithm stops.

5 Numerical Experiments

In our experiment, the time horizon $T = 180$ and other supply chain parameters are listed in Table 1.

Table 1. Parameters of the supply chain

	hOf	*hOv*	*hI*	*hU*	Lead Time(day)
The retailer	10	0.05	0.01	10	2
The supplier	30	0.02	0.005	30	5

In the PSO, number of the particles *pt_num* = 100, ω= 0.1, c_1= c_2 = 2 and the generations of evolution *max_gen* = 300.

The larger number of sampling sequence is, the better result will be obtained, but the time cost will increase too. So we choose N = 30 here to make a balance between the solution accuracy and time cost.

The uncertain demand is supposed to follow a normal distribution N(80, 16). In order to describe the uncertainty of demand, two different approaches are employed. One is robust optimization approach which use the random sampling sequences following the normal distribution N(80, 16) as the customers' demand to calculate the total cost of supply chain; the other is traditional optimization approach which use the mean of the normal distribution 80 to calculate the total cost. Both PSO-based approaches run 20 times and the obtained decision variables are given in Table 2 and Table 3 respectively.

In order to evaluate the solutions, each solution is tested by 5000 sampling sequences which follow the normal distribution N(80, 16) and each sequence corresponds to a total cost. Then we calculate the mean and the standard deviation of these 5000 total costs which are denoted as TC_{exp}, TC_{std}, respectively. They are also listed in the tables. The number of evaluations required to converge denoted as N_e is also given in the last column.

Table 2. Experimental results obtained by the PSO based robust optimization approach

No.	n_r	n_q	q	r	TC_{exp}	TC_{std}	N_e
1	3	0	493	245	2502.78	40.35	204000
2	15	8	246	246	5113.31	67.64	96000
3	9	7	246	246	4325.20	67.60	264000
4	2	0	495	246	2427.32	32.59	126000
5	6	5	247	247	3631.87	51.71	147000
6	4	1	251	249	2625.21	27.76	210000
7	4	1	251	248	2625.82	28.39	129000
8	12	6	246	246	4367.39	69.20	153000
9	9	2	248	248	3176.79	41.65	318000
10	2	0	494	246	2425.80	35.13	165000
11	7	5	246	246	3718.02	66.46	288000
12	2	0	494	247	2425.60	33.67	156000
13	2	0	494	245	2428.15	38.76	285000
14	3	0	495	244	2508.76	39.14	201000
15	6	1	250	248	2704.21	36.63	129000
16	9	3	246	246	3383.15	67.62	123000
17	8	1	248	248	2884.40	40.69	141000
18	2	0	494	247	2425.89	33.81	171000
19	4	1	250	248	2629.71	36.53	171000
20	2	0	493	247	2426.02	36.37	207000
average					3037.77	44.59	184200

Table 3. Experimental results obtained by the PSO based traditional optimization approach

No.	n_r	n_q	q	r	TC_{exp}	TC_{std}	N_e
1	14	1	240	225	4378.55	459.54	6200
2	2	0	480	230	3121.26	407.09	10800
3	4	1	240	227	3569.22	456.66	7600
4	2	0	480	238	2675.86	218.66	4600
5	2	0	480	228	3210.11	425.42	5300
6	10	5	240	230	4804.67	458.98	4700
7	13	7	240	225	5517.37	455.57	8100
8	2	0	480	234	2875.54	318.36	4400
9	11	6	240	230	5137.75	458.09	5200
10	8	1	240	238	3774.40	461.95	5300
11	3	2	240	231	3829.19	464.24	4900
12	9	3	240	237	4311.66	466.97	5900
13	10	4	240	218	4622.90	454.65	17100
14	14	1	240	231	4380.37	459.63	21700
15	10	1	240	191	4376.62	489.52	11300
16	12	1	240	237	4138.68	467.87	8600
17	2	0	480	199	3509.27	445.63	6600
18	9	3	240	226	4309.87	458.38	7700
19	11	4	240	240	4644.05	461.49	5500
20	9	3	240	237	4306.66	470.19	4900
average					4074.70	437.94	7820

From Table2 and Table 3, we can find that the mean of TC_{exp}, TC_{std} obtained by traditional optimization approach are larger than that obtained by robust optimization approach. It shows that the proposed robust optimization approach can produce a collaboration decision scheme that is robust to the effect of demand uncertainty, but more evaluations are required by the robust approach to converge.

6 Concluding Remarks

In this paper, a PSO based robust optimization approach is proposed for the problem of supply chain collaboration with uncertain demand. The approach can obtain a robust collaboration decision scheme, which is insensible to the demand uncertainty that follows various probability distributions. The proposed approach requires more computation time than the traditional method. The more demand sampling sequences are used, the better result can be obtained, but more time will be spent. Therefore, we need to set a balance of accuracy and time cost according to the practical requirements.

The standard PSO is employed in the robust optimization algorithm and its effectiveness has been proved by numerical experiments. However, the standard PSO has the shortage of easily sticking at local optimum. Many studies have been made to improve its performances. Our future work is to develop the improved PSO to further enhance the accuracy of the solution, reduce the time complexity and compare with other optimization algorithms.

Acknowledgment

This research is supported by the National High Technology Research and Development Program ("863"Program) of China (Grant No. 2009AA04Z120) and the National Natural Science Foundation Project of China (Grant No. 60504028).

References

1. Fung, R.Y.K., Chen, T.: A multiagent supply chain planning and coordination architecture. Int. J. Adv. Manuf. Technol. 25(7/8), 811–819 (2005)
2. Li, X., Wang, Q.: Coordination mechanisms of supply chain systems. Eur. J. Oper. Res. 179(1), 1–16 (2007)
3. Barbarosoglu, G., Ozgur, D.: Hierarchical design of integrated production and 2-echelon distribution system. Eur. J. Oper. Res. 118, 464–484 (1999)
4. Ozdamar, L., Yazgac, T.: A hierarchical planning approach for a production-distribution system. Int. J. Prod. Res. 37(16), 3759–3772 (1999)
5. Darwish, M.A., Odah, O.M.: Vendor managed inventory model for single-vendor multi-retailer supply chains. Eur. J. Oper. Res. 204, 473–484 (2010)
6. Kim, J.U., Kim, Y.D.A.: Lagrangian relaxation approach to multi-period inventory/distribution planning. J. Oper. Res. Soc. 51(3), 364–370 (2000)
7. Kang, J.-H., Kim, Y.-D.: Inventory replenishment and delivery planning in a two-level supply chain with compound Poisson demands. Int. J. Adv. Manuf. Technol. 49 (9/12), 1107–1118 (2009)
8. Schmitt, A.J., Snyder, L.V., Shen, Z.-J.M.: Inventory systems with stochastic demand and supply: Properties and approximations. Eur. J. Oper. Res. 206, 313–328 (2010)

A Multi-valued Discrete Particle Swarm Optimization for the Evacuation Vehicle Routing Problem

Marina Yusoff[1], Junaidah Ariffin[2], and Azlinah Mohamed[1]

[1] Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA
40450 Shah Alam, Selangor
Malaysia
{marinay,azlinah}@tmsk.uitm.edu.my
[2] Flood-Marine Excellence Centre
Faculty of Civil Engineering
Universiti Teknologi MARA
40450 Shah Alam, Selangor
Malaysia
junaidahariffin@yahoo.com

Abstract. An optimal evacuation route plan has to be established to overcome the problem of poor coordination and uneven distribution of vehicles before or during disaster. This article introduces the evacuation vehicle routing problem (EVRP) as a new variant to the vehicle routing problem (VRP). EVRP is a process of moving vehicles from a vehicle location to the potentially flooded area (PFA), and from PFA to relief center using a number of capacitated vehicles. This paper examines the application of a multi-valued discrete particle swarm optimization (DPSO) for routing of vehicles from vehicle location to PFA. A solution representation is adopted and modified from the solution of the shortest path problem (SPP) to accommodate this problem. Experimental results were tested based on the objective function of finding a minimum total travelling time using datasets from a flash flood evacuation operation. DPSO was found to yield better results than a genetic algorithm (GA).

Keywords: Keywords-discrete particle swarm optimization; evacuation route plan; evacuation vehicle routing problem; potentially flooded area; vehicle routing problem.

1 Introduction

During flood evacuation, the determination of the optimal evacuation route for moving people to safety is made more difficult by the unavailability of information required by the agencies responding to the floods. This inadequacy in flood response triggered the National Security Council of Malaysia to formulate sustainable emergency procedures for management of evacuees and properties. Nevertheless, these guidelines on disaster management have limited advantages (M. Omar, personal communication, November, 2007). They merely serve as a very generic manual

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 182–193, 2011.
© Springer-Verlag Berlin Heidelberg 2011

guiding response to disaster. In all circumstances of evacuation planning, immediate response is crucial as time is the decisive factor. Therefore, immediately after warning has been issued an emergency evacuation route plan should be generated considering the safe routes for evacuees and allocation of vehicles. An optimization algorithm is required to help in producing an optimal emergency evacuation plan to help local authorities and related agencies making prompt and informed decisions. A few optimization algorithms have been developed in other countries for evacuation planning in several types of natural disaster, namely Capacity Constrained Route Planning [1-3], A* [3], Flip High Flip Edge [4], greedy heuristic [5-6], Bottleneck Relief Heuristic [5-6], BEST [7], SP-TAG [7], and multi-ant colony system [8]. Some of them have demonstrated good performance in optimizing evacuation planning, but they do not focus on the capacity of vehicles during routing. Thus, this paper introduces the evacuation vehicle routing problem (EVRP) as a new variant to the vehicle routing problem (VRP) since it has same core process as VRP. EVRP considers the routing of vehicles that have been assigned for a particular number of people. Prior to solving the EVRP, a list of assigned vehicles with their capacities is generated using DPSO-VAP [9]. It should be noted that in finding solutions to EVRP, it is unnecessary to find separately the solution of the shortest path problem (SPP) since SPP is a basic process of a VRP.

In this paper, we consider the static routing of capacitated vehicles from vehicle location to PFA to find a minimum total travelling time using multi-valued discrete particle swarm optimization (DPSO). PSO has shown significant results for solving SPP and outperformed genetic algorithm (GA) [10]. A range of random discrete priority values (PVs) that represents all of the nodes in a network graph offers a 95% optimal solution for PSO in this research. A further reason for using DPSO is its successful performance in the variant of VRP that is similar to EVRP. It has been observed that EVRP is closely related to the capacitated vehicle routing problem (CVRP), primarily in its handling capacity constraints [11-14]. In particular, EVRP deals with routing of a number of capacitated vehicles to PFA, whereas CVRP deals with the delivery of goods to customers. Like EVRP, CVRP assumes that each customer is served by exactly one vehicle without exceeding the capacity constraints of each vehicle.

Several optimization algorithms have been employed [11-14] for solving CVRP. For example, GA with local search is applied in [11] and DPSO with binary position and a hybrid of DPSO-SA in [12]. In general, they obtained an effective result in terms of processing time with no assurance for optimal results. In addition, with the embedment of the two solution representations for CVRP in PSO, namely SR-1 and SR-2, it has been shown that the PSO gives a good quality solution [15]. These representations use a real value for the particle position representation comprising customer priority and vehicle priority. SR-1 finds a sequence of routes for each vehicle which routes the vehicle to the customer, based on the Euclidean distance between customer and vehicle reference point, where as SR-2 incorporates the coverage of radius based on vehicle orientation point for vehicle routing. Local improvements were added to change the customer route direction to directly improve the routing cost. Although this solution has seemed possible for a search on optimal EVRP results, there are still some limitation of information to generate PFA's priority and

vehicle's priority value. Consequently, the EVRP solution is based on the problem formulation and a discrete particle representation in Section 3.

This paper is organized as follows. Section 2 reviews the PSO algorithm. Section 3 presents the problem formulation and solution representation. The DPSO algorithm is discussed in Section 4, and Section 5 explains the computational results and discussion. Finally, Section 6 concludes the paper and addresses some future work.

2 Particle Swarm Optimization

PSO was introduced by Kennedy and Eberhart in the mid-1990s [16]. A population-based stochastic approach grouped under swarm intelligence [17], it is used to solve continuous and discrete problems. PSO indicates the velocity and position of particles in a multi-dimensional space. By updating both velocity and position, a feasible solution is achieved. The fitness values comprise G_{best} and P_{best} , which derive from the simulated behaviour of a group of particles [18]. PSO is able to explore regions of the search space and exploit the search to refine a feasible solution. These search strategies are influenced by the parameters acceleration constant and inertia weight [19-20]. Equation 1 and equation 2 present the velocity and position formulas for the canonical PSO, respectively.

$$V_{id(new)}=W\ x\ V_{id(old)} + C_1\ x\ r\ x(P_{best} - X_{id(old)} + C_2\ x\ r\ x(G_{best}-X_{id(old)}) \quad (1)$$

$$X_{id(new)}=X_{id(old)} + V_{id(new)} \quad (2)$$

where $V_{id(new)}$ and $V_{id(old)}$ are the new and old velocities of particle i, respectively; $X_{id(old)}$ and $X_{id(new)}$ are the old and new particle positions respectively; and W is the inertia weight. C_1 and C_2 are the acceleration constant parameters, r is the random function in the range of [0,1], P_{best} is the personal best of the i^{th} particle, and G_{best} is the best position derived from all particles in the swarm.

PSO has been shown to be useful to solve such types of problems as the travelling salesman problem [20-21] and vehicle routing [12]. PSO is easy to implement and is computationally efficient [22-23]. Modification of PSO have been developed to the performance of PSO for various types of problems [24-25] and across standard benchmark datasets [21][24][26]. For example, the canonical PSO applies inertia weight in updating velocity to simulate the social behavior of birds. After two years of PSO development, the research on a discrete problem had concentrated on discrete binary PSO begun by Kennedy and Eberhart [27]. Kennedy and Eberhart proposed a new way of updating the position of particles to accommodate a binary representation. This solution was then improved in several studies based on a benchmark [28-29] and a real-world situation [30-31] using DPSO algorithm. When compared to other optimization methods, the performance of DPSO has been found to be competitive with a genetic algorithm [30], demonstrating the promise of DPSO, with its global search capability and local exploitation.

3 Problem Formulation and Solution Representation

This paper focuses on the EVRP that involves routing of a number of vehicles from a vehicle location to a single destination (i.e., only one PFA). EVRP addresses the objective function to find a minimum total traveling time for all the capacitated vehicles from the vehicle location to the PFA. This problem is mathematically formulated based on with terminology used in previous research on VRP [13][32]. The problem can be formally defined as follows: Let $G = (N, E)$ be a weighted directed graph. Define $N = \{N_0, N_2, N_3, ... N_n\}$. N_0 represents a vehicle location and N_n is a destination node (PFA). E is the set of edges. C_{ij} are matrices representing travelling cost of traversing from i to j. For each edge $(i, j) \in E$, a distance $d_{ij} \geq 0$ and travel time $t_{ij} \geq 0$, is a non negative integers. V_{ij} is the set of all vehicles that are able to move from node i and j, where $V = \{V_1, V_2, ..., V_k\}$. The capacity of vehicle, $c \in V$ is denoted as V_C. The decision variable X_{ijk} is a binary variable which has the value of 1 if vehicle k travels from node i to node j, otherwise 0. The objective function is to find a minimum total travelling time for all vehicles from N_0 to N_n. The EVRP is mathematically formulated as shown below:

$$\text{Min} \sum_{k \in K} \sum_{(i,j) \in N} C_{ij} X_{ijk} \tag{3}$$

subject to

$$\sum_{k \in K} \sum_{i \in N} \sum_{j \in N} X_{ijk} = 1 \tag{4}$$

$$\sum_{j \in N} X_{ijk} - \sum_{j \in N} X_{jik} = 0 \tag{5}$$

$$X_{ijk} \in \{0,1\} \tag{6}$$

Constraints (4) and (5) ensure that all vehicles travelled. Constraint (6) is the set of bound decision variables. The solution representation for EVRP is adopted from the work of Mohemmed et al [10] because of its good performance in SPP. To accommodate the EVRP, we enhance this representation taking into account a number of capacitated vehicles. However, the use of PV that represents each node is maintained. As shown in Fig. 1, the representation of a particle consists of an array of PVs that is assigned to each of the capacitated vehicles. In other words, each particle comprises a matrix of PV_n x V_m, that is, the matrix of a particle would depend on the total number of nodes and the number of generated vehicles.

PV_0	PV_1	PV_2	PV_3	...	PV_{n-1}	PV_n	V_1
PV_0	PV_1	PV_2	PV_3	...	PV_{n-1}	PV_n	V_2
...	...	...	...	...	...	...	.
PV	PV_1	PV_2	PV_3	...	PV_{n-1}	PV_n	V_m

Fig. 1. A particle comprises a matrix of PV_n x V_m

Each vehicle traverses from its vehicle location (N_0), trying to find a valid path to the PFA (N_n). If a path, for example from N_0 to N_1, is a valid path, then, the travelling time for the vehicle is calculated. The travelling time depends on the distance for the valid path and the standard travelling speed of the vehicle, as calculated using the following equation.

$$t_v = d / ts_v \tag{7}$$

where t_v is travelling time for each vehicle for the valid path travelled, d is a distance between two nodes (edge), and ts_v is a standard travelling speed for each vehicle. The total travelling time for each of vehicle travelled through valid path is then calculated using Equation 8.

$$tt_v = \sum_1^m t_v \tag{8}$$

where tt_v is the total travelling time for all vehicles travelling through all valid paths. Total travelling time for all vehicles is based on the number of vehicles travelling on the valid path, as calculated using Equation 9.

$$tt_{vs} = \sum_1^m tt_v \tag{9}$$

tt_{vs} is presented as the P_{best} calculated for each population of particle. The total travelling time might be different because the number of vehicles travelling depends on the vehicles generated using myDPSO-VAP [9] and their standard travelling speed. The next section explains the solution representation that is adopted into the DPSO algorithm.

4 Discrete Particle Swarm Optimization

The solution representation described in Section 3 is implemented in the DPSO as shown in Fig. 2. This algorithm is similar to that found in Mohemmed et al [10]. The number of vehicles traversing from the starting node N_0 (vehicle location) to N_n (PFA) is also considered. The algorithm starts with the normal process of PSO. Steps 2 and 3 initialize the number of populations and the coefficient values C_1 and C_2, respectively. Step 4 performs the initialization of PVs and velocities. Step 5 retrieves vehicle's information which includes the vehicle id, vehicle capacity, and its standard travelling speed. Steps 6 through 11 follow the same step as described in Mohemmed et al [10]. Then, P_{best} and G_{best} are calculated. In summary, steps 2 through 12 yield an initial solution, which is followed by an iteration process that begins at step 13 until 27 until the final iteration is achieved with all vehicles arriving at the destination.

Each particle is updated with a new velocity and position value (PV) using Equation 1 and Equation 2 at step 14 until 17. The new velocity and new position value are in the form of a multi discrete value (positive integer). Step 18 executes path validation for each of the vehicles. Step 19 will take place in the event of a vehicle arriving at destination. If one vehicle arrives at the destination, other vehicles can use the PV

that was used by the vehicle that has arrived, ensuring that all vehicles would be able to arrive at their respective destinations. This condition can be seen as the probability of achieving an optimal or sub-optimal result for an iteration of a particle, if the vehicle travelled for the minimum total travelling time. DPSO's random population of particles and its ability to exploit and explore enhance the possibility of obtaining the optimal result. Finally, steps 20 to 26 calculate $P_{best(new)}$, $G_{best(new)}$ and the condition for the selection of the best current fitness for each iteration.

```
1:  Begin
2:    Initialize number of particle's population
3:    Declare C1 and C2
4:    Initialize PVs, Vmin and Vmax for all particles in random
5:    Retrieve vehicle's information from [9]
6:    For each vehicles
7:      Construct a path
8:        If it is a valid path
9:          Calculate tv
10:   else
11:       Return fitness value as 0
12: Calculate Pbest(old) and Gbest(old)
13: Do
14:   For each particle
15:       Calculate V(new) using equation (1)
16:       Calculate PV(new) using equation (2)
17:       Update PVs
18:       Perform step 6 until 11
19:         If there is a vehicle arrive at destination, other vehicles applies the PV used by this vehicle
20:   Calculate Pbest (new)
21:   Calculate Gbest (new)
22:    If (Gbest (new) > Gbest(old))
23:       Assign Gbest(old) as the best current fitness
24:   If (Gbest (new) =< Gbest(old))
25:      Gbest(old))= Gbest (new)
26:     Assign Gbest(new) as the best current fitness
27:   While (maximum iteration is achieved or all vehicles arrived at destination)
28: End
```

Fig. 2. DPSO algorithm

5 Computational Result and Discussion

This section presents the computational results to allow evaluation of the performance of the DPSO algorithm in the application of a multi discrete particle position in EVRP. The DPSO algorithm was implemented in JAVA and run on a PC with an Intel Core 2 CPU (3.0 GHz) and 2GB memory. To verify the proposed algorithm, a GA with the application of the same solution representation as DPSO is used. In this case, the GA with one point crossover and the same number of population was compared to DPSO. The following subsections introduce the experimental setup and the datasets, and present and discuss the results using DPSO and GA.

5.1 Experimental Setup

Table 1 is the list of parameters for the computational experiments.

Table 1. List of parameters

Parameter	Value	Parameter	Value
PV_{max}	-100 [10]	*Initial* V_{min}	-10 [10]
PV_{min}	100 [10]	*Initial* V_{max}	10 [10]
C_1	2.05 [19]	Inertia Weight, w	0.12 [20]
C_2	2.05 [19]	Stopping condition	Until all vehicles are arrived destination or certain 200 iterations

The stopping condition is based on all vehicles are arrived destination or 200 iterations. Datasets were taken from data generated in dealing with a flash flood in Malaysia's Kota Tinggi district in December 2006 and 2007. The assigned vehicles were generated using algorithm in [9], which involves the information of vehicles; vehicle id, destination node for each vehicle, number of people assigned to the particular vehicle, and travelling speed for each vehicle. Routes from vehicle location to PFA are indicated by source nodes (original vehicle location), nodes, edges, and destination node (PFA). All of the routes are transformed into graph abstraction. The graph is then transferred into an adjacency matrix for easy transformation into the algorithm. Table 2 shows the datasets for routing comprising the number of nodes, the total number of people that need to be evacuated and the number of vehicles generated.

Table 2. List of datasets from flash flood evacuation in 2006 and 2007 for a single PFA

Datasets	Number of nodes	Number people	Number of the generated vehicles
VR1_06	12	5853	600
VR2_06	35	26	3
VR3_06	12	1215	154
VR4_06	21	448	52
VR5_06	25	524	59
VR6_06	36	1429	224
VR1_07	12	1620	253
VR2_07	21	620	94

5.2 Comparison of Solutions Using DPSO and GA

The performance of DPSO and GA are analyzed based on the objective function to find a minimum total traveling time for all the capacitated vehicles from the vehicle location to the PFA. The comparisons involve two aspects: total travelling time for all vehicles from vehicle location to PFA and processing time. Table 3 compares the results of VR1_06 based on the fitness value (in hours) and processing time (in seconds) for DPSO and GA. The total travelling time (tt_{vs}) is obtained from the total of 600 vehicles which successfully arrived at the PFA. As can be seen in Table 3, with 10 populations, the fitness value of GA is about 0.039 seconds less than that of DPSO, but GA requires a longer processing time than does DPSO. In terms of number of iterations used for this population, GA performs only one iteration while DPSO performs 23 iterations. However, most interestingly, DPSO outperformed GA for 30 populations in terms of processing time with about 0.296 seconds while GA requires 0.319 seconds. DPSO for 50 populations in terms of total travelling time of about 8.797 hours using only one iteration while GA requires 9.136.

Contrary to expectations, neither DPSO nor GA produced any result after 200 iterations for VR2_06, VR4_06, VR5_06, and VR6_06. This failure may be accounted for by the fact that the multi-valued PVs assigned to each node were unable to determine valid paths. These datasets used a greater number of nodes than that of VR1_06. This shows that the particles utilize more search space compared to VR1_06. The next comparison highlights the results of the VR3_06. As can be seen in Table 3, both DPSO and GA performance obtained the same fitness for 50 populations using this dataset. However, DPSO took 0.002 second longer than GA. As can be seen in the table, GA gives less quality solution. GA requires 0.072 seconds of processing time using 50 populations while DPSO requires 0.070 seconds using 10 populations. In particular, DPSO demonstrates better performance than GA, when using less than 40 populations. The result indicates that the use of multi-valued PVs in DPSO and GA provides a minimum total travelling time to destination for this dataset.

Table 3. Performance of DPSO and GA using VR1_06 and VR3_06

P	Dataset VR1_06						Dataset VR3_06					
	DPSO			GA			DPSO			GA		
	tt_{vs}	*PT (s)*	*iter*	tt_{vs}	*PT (s)*	*Iter*	tt_{vs}	*PT (s)*	*iter*	tt_{vs}	*PT (s)*	*Iter*
10	9.165	0.343	23	9.136	0.415	1	6.459	0.070	2	-	-	200
20	9.136	0.328	2	9.136	0.313	1	6.459	0.072	7	6.459	0.140	1
30	**9.136**	**0.296**	2	9.136	0.319	1	6.459	0.070	23	-	-	200
40	9.136	0.390	1	9.136	0.313	1	6.459	0.080	6	6.459	0.095	1
50	8.797	0.452	1	9.136	0.378	1	6.459	0.074	12	6.459	0.072	2

* P - number of population, PT - *p*rocessing time, iter - number of iteration

Table 4 and 5 shows the results of the datasets of a single PFA concentration on 30 populations of particles, 30 experiments and based on the iteration up to 200 or until all vehicles arrived at the PFA. The selection of 30 populations is based on the suggestion from Mohammed et al [10]. The average of the total travelling time and processing time is based on 30 experiments. As shown in Table 4, it is apparent that DPSO outperformed GA of about 0.51% lesser in its average of fitness value. However, the average of processing time of DPSO is shown competitive to GA. In contrast, no results found for GA, using dataset VR3_06. The DPSO provides a consistent of 6.459 hours of its total travelling time, with less than one second of processing time. In addition, DPSO confirm gives a better fitness value to EVRP. The evident is shown in Table 5 showing that DPSO outperformed the GA for both dataset VR1_07 and VR2_07. Overall, DPSO and GA consume in average less than one second of processing time.

Table 4. Comparison measure of DPSO and GA using VR1_06 and VR3_06 based on average, min, max, and standard deviation

	Dataset VR1_06						Dataset VR3_06					
	DPSO			GA			DPSO			GA		
	tt_{vs}	*PT (s)*	*Iter*	tt_{vs}	*PT (s)*	*iter*	tt_{vs}	*PT (s)*	*iter*	tt_{vs}	*PT (s)*	*iter*
Avg	9.129	0.468	2	9.176	0.460	1	6.459	0.253	2	-	-	200
Min	8.797	0.312	1	8.797	0.328	1	6.459	0.129	1	-	-	200
Max	9.880	0.843	15	9.880	0.671	2	6.459	0.365	32	-	-	200
Std Dev	0.176	0.136	2.609	0.200	0.107	0.498	0.000	0.043	5.721	-	-	-

* P - number of population, PT - processing time, iter - number of iteration

Table 5. Comparison measure of DPSO and GA using VR1_07 and VR2_07 based on average, min, max, and standard deviation

	Dataset VR1_07						Dataset VR2_07					
	DPSO			GA			DPSO			GA		
	tt_{vs}	*PT (s)*	*Iter*	tt_{vs}	*PT (s)*	*iter*	tt_{vs}	*PT (s)*	*iter*	tt_{vs}	*PT (s)*	*iter*
Avg	3.839	0.199	8	3.844	0.196	2	**2.046**	3.146	14	-	-	200
Min	3.704	0.140	1	3.704	0.140	1	1.990	3.093	4	-	-	200
Max	4.160	0.327	101	4.160	0.280	2	2.369	3.296	91	-	-	200
Std Dev	0.078	0.051	20.287	0.074	0.041	0.498	0.114	0.045	24.103	-	-	0

* P - number of population, PT - processing time, iter - number of iteration

This study produced results, which corroborate the findings of Mohemmed et al [10] for SPP as it was offered good solution. The findings confirmed that DPSO proved better than GA in getting a minimum total travelling time. The use of multi-valued discrete particle position is observed to have successfully achieved optimal results for a small number of datasets. In addition, the processing time of less than one second is acceptable, especially for the VR1_06 dataset, where 600 vehicles are traversed through all nodes to their destinations. Based on the interesting findings provided by DPSO, it can be illustrated that several valid paths were able to determine using 30 populations of particles, grants a higher possibility of using less travelling time for the vehicles travelled from vehicle location to a single PFA. With the high possibility of getting valid node, the best solution would become faster and lead to the fast convergence due to the less search space. This is confirmed by was mentioned in the literature review that the DPSO has a capability of finding better solution and fast convergence compared to the GA.

6 Conclusion and Recommendation

A solution representation for EVRP has been presented using both DPSO and GA. DPSO was found to achieve better solution quality in terms of total travelling time

and processing time compared to GA. The random multi-value of PVs and the formulation of updating velocity and PVs used in DPSO support this solution. In future investigations, it might be possible to find mean of limiting the movement of particles, most probably with the decomposition of graph. This is to ensure at least one vehicle can traverse from vehicle location until destination using a valid path. In addition, more experiments are required with the consideration of different parameter values and different size of routing dataset.

Acknowledgment

This study has been made possible under the support of the Ministry of Science and Technology Malaysia through the Science Fund and University Technology MARA.

References

1. Lu, Q., Huang, Y., Shekhar, S.: Evacuation Planning: A Capacity Constrained Routing Approach. In: Chen, H., Miranda, R., Zeng, D.D., Demchak, C.C., Schroeder, J., Madhusudan, T. (eds.) ISI 2003. LNCS, vol. 2665, pp. 111–125. Springer, Heidelberg (2003)
2. Lu, Q., George, B., Shekhar, S.: Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. In: Anshelevich, E., Egenhofer, M.J., Hwang, J. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 291–307. Springer, Heidelberg (2005)
3. Lu, Q.: Capacity constrained routing algorithms for evacuation route planning. PHD Thesis, University of Minnesota (2006)
4. Kim, S., Shekhar, S.: Contraflow network reconfiguration for evacuation planning: a summary of results. In: 13th Annual ACM International Workshop on Geographic Information Systems, pp. 250–259 (2005)
5. Shekhar, S., Kim, S.: Contraflow transportation network reconfiguration for evacuation route planning. Technical Report, Mn/DOT 2006-21, Department of Computer Science and Engineering, University of Minnesota (2006)
6. Kim, S.: Contraflow network reconfiguration using evacuation route planner. PHD Thesis, University of Minnesota (2007)
7. George, B., Kim, S., Shekhar, S.: Spatio-temporal network databases and routing algorithms: A summary of results. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 460–477. Springer, Heidelberg (2007)
8. Zong, X., Xiong, S., Fang, Z., Li, Q.: Multi-ant colony system for evacuation routing problem with mixed traffic flow. In: Congress on Evolutionary Computation, pp. 1–6 (2010)
9. Yusoff, M., Ariffin, J., Mohamed, A.: Solving Vehicle Assignment Problem Using Evolutionary Computation. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 523–532. Springer, Heidelberg (2010)
10. Mohemmed, A.W., Sahoo, N.C., Geok, T.K.: Solving shortest path problem using particle swarm optimization. Applied Soft Computing 8(4), 1643–1653 (2008)
11. Lin, S.-W., Ying, K.-C., Lee, Z.-J., Hsi, F.-H.: Applying simulated annealing approach for capacitated vehicle routing problems. In: IEEE International Conference on Systems, Man and Cybernetics (SMC), Taipei, pp. 639–644 (2006)

12. Chen, A.-l., Yang, G.-k., Wu, Z.-m.: Hybrid Discrete Particle Swarm Optimization Algorithm for Capacitated Vehicle Routing Problem. Journal of Zhejiang University Science 7, 607–614 (2006)
13. Zhishuo, L., Yueting, C.: A hybrid ant colony algorithm for capacitated vehicle routing problem. In: IEEE International Conference on Systems, Man, and Cybernatics, pp. 3907—3911 (2006)
14. Marinakis, Y., Marinaki, M.: A particle swarm optimization algorithm with path relinking for the location routing problem. Journal of Mathematical Modelling and Algorithms 7, 59–78 (2008)
15. Ai, T.J., Kachitivhyanukul, V.: Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. Computers and Industrial Engineering 56, 380–387 (2009)
16. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942—1948 (1995)
17. Engelbrecht, A.P.: Computational intelligence: An Introduction, 2nd edn., p. 9. John Wiley & Son, West Sussex (2007)
18. Guner, R., Sevkli, M.: A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. Journal of Artificial Evolution and Applications, 1–9 (2008)
19. Engelbrecht, A.P.: Computational intelligence: An Introduction, 2nd edn., pp. 306–309. John Wiley & Son, West Sussex (2007)
20. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation. IEEE Service Center, Piscataway (1999)
21. Zhong, W.-l., Zhang, J., Chen, W.-n.: A novel discrete particle swarm optimization to solve traveling salesman problem. In: IEEE Congress on Evolutionary Computation (CEC), pp. 3283—3287 (2007)
22. Li, X., Tian, P., Hua, J., Zhong, N.: A hybrid discrete particle swarm optimization for the traveling salesman problem. In: Wang, T.-D., Li, X., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 181–188. Springer, Heidelberg (2006)
23. Eberhart, R. C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS), pp. 39–43 (1995)
24. Veeramachaneni, K., Osadciw, L., Kamath, G.: Probabilistically driven particle swarms for optimization of multi valued discrete problems: design and analysis. In IEEE Swarm Intelligence Symposium (SIS), pp. 141—149 (2007)
25. AlHajri, M. F., AlRashidi, M. R., El-Hawary.: A novel discrete particle swarm optimization algorithm for optimal capacitor placement and sizing. In: Electrical and Computer Engineering (CCECE), pp. 1286–1289 (2007)
26. Al-Kazemi, B., Mohan, C.K.: Discrete Multi-Phase Particle Swarm Optimization. In: Information Processing with Evolutionary Algorithms. Advanced Information and Knowledge Processing, pp. 305–327. Springer, Heidelberg (2005)
27. Kennedy J, Eberhart R.C.: A Discrete Binary Version of the Particle Swarm Algorithm. In IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL, USA, vol. 5, pp. 4104–4108 (1997)
28. Eberhart R.C., Shi, Y.: Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. Evolutionary Computation. In Proceedings of the Congress on Evolutionary, vol. 1, pp. 84–88 (2000)

29. Bui, L.T., Soliman, O., Abbass, H.A.A.: Modified Strategy for the Constriction Factor in Particle Swarm Optimization. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS (LNAI), vol. 4828, pp. 333–344. Springer, Heidelberg (2007)
30. Gong, T., Tuson, L.: Particle swarm optimization for quadratic assignment problems - a forma analysis approach. Computational Intelligence Research 4, 177–185 (2008)
31. Gao, F., Cui, G., Zhao, Q., Liu, H.: Application of improved discrete particle swarm algorithm in partner selection of virtual enterprise. Computer Science and Network Security 6, 208–212 (2006)
32. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Computers and Operations Research 34, 2403–2435 (2007)

A NichePSO Algorithm Based Method for Process Window Selection

Wenqi Li[1,2], Yiming Qiu[1], Lei Wang[1], and Qidi Wu[1]

[1] School of Electronics and Information Engineering, Tongji University, Shanghai, 200092, China
[2] Analog Devices Technology (Shanghai) Co. Ltd., Shanghai, 200021, China
robert.li@analog.com, hans.qiu@keb.cn, wanglei_tj@126.com

Abstract. Process parameter window selection in semiconductor manufacturing field is usually the problem to find out the ranges of input parameters that meet production requirements, which requires allocating optima of a multimodal function efficiently. To achieve good results under the conditions of multimodal model and process control requirement, a NichePSO algorithm based method for parameter window selection is presented in this paper. Both simulation results and production validation data indicate it is an effective method for process parameter window selection.

Keywords: NichePSO, Process Optimization, Parameter Window Selection.

1 Introduction

Statistical Process Control (SPC) has been widely applied to semiconductor manufacturing field to monitor process parameters and detect process abnormality. The data points of a stable and capable process vary randomly between Upper Control Limit (UCL) and Lower Control Limit (LCL), and it will be recommended to be adjusted once any data point violates the SPC control rule. To secure a wide enough range for each process parameter, the Upper Specification Limit (*USL*) and Lower Specification Limit (*LSL*) of the parameter must be wider than *UCL* and *LCL*. Normally process control in semiconductor production requires sigma level to be 3 or 6, and process capability index *Cpk* should be equal to or greater than 1.67[5]. Control chart example in figure 1 presents the relations between the indicators:

$$LCL = \overline{X} - 3\sigma \quad . \tag{1}$$

$$UCL = \overline{X} + 3\sigma \quad . \tag{2}$$

$$Cpk = \min\left(\frac{USL - \overline{X}}{3\sigma}, \frac{\overline{X} - LSL}{3\sigma}\right) \geq 1.67$$

$$\text{or} \quad \begin{cases} LSL \leq \overline{X} - 1.67 \times 3\sigma \\ USL \geq \overline{X} + 1.67 \times 3\sigma \end{cases} . \tag{3}$$

where

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad , \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2} \quad .$$

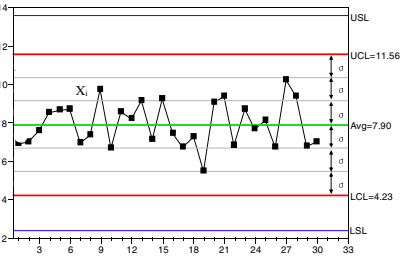


Fig. 1. Control chart example

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 194–202, 2011.
© Springer-Verlag Berlin Heidelberg 2011

LSL and *USL* of each parameter are defined during process development stage according to (3), so the width of the process window *W* should be wide enough:

$$W \geq USL - LSL \geq 1.67 \times 3\sigma + 1.67 \times 3\sigma = 10.02\sigma \quad . \tag{4}$$

In some of real cases, output parameter Q is a multimodal function of multiple input parameters P_i, and it must meet the process requirement of $Q>R$, or $Q<R$ or $R_1<Q<R_2$. This paper takes the scenario of $Q>R$, then:

$$\begin{cases} Q = Q(P_1, P_2, \ldots P_n) \\ Q > R \end{cases} \quad . \tag{5}$$

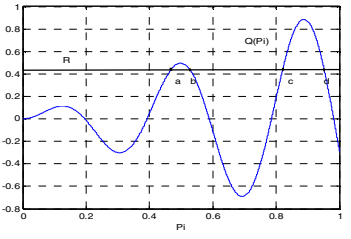


Fig. 2. Input, output parameters and process requirement

Each input parameter window must satisfy (4) and (5) and be as wide as possible, which is equivalent to find out all common points of Q and R, and figure out the widest portion that meets the both equations, such as the portion between c and d in figure 2. Usually, the function of Q is unknown and engineer has to define input parameter window based on experience and experiment data, which makes the problem more complex, so an effective method to select parameter window will be very helpful.

Particle Swarm Optimization (PSO) has been successfully applied to solve unimodal optimization problem and has been proven as an effective algorithm, but it can find one optimum only and may not converge if there are multiple optima with equal or similar values. Niching Particle Swarm Optimizer (NichePSO) algorithm that developed by R. Brits can find out all optima including both global optimum and local optima, which is an effective tool to solve multimodal optimization problem. NichePSO algorithm is introduced and a parameter window selection method based on it is presented in this paper. Section 2 gives a brief overview of PSO and NichePSO algorithms, and section 3 provides a NichePSO algorithm based method for parameter window selection. Simulation results and production validation data of the method are summarized in section 4 and 5.

2 Niching Particle Swarm Optimizer

2.1 Particle Swarm Optimization

Particle swarm optimization (PSO) algorithm was developed by J, Kennedy and R. Eberhart in 1995 when it was used to study the food searching behaviors of bird flocks[1],[3]. Each particle in PSO algorithm is regarded as a point with zero size and mass, and it updates its position based on its current velocity *V*, the best position found by itself (p_{best}) and the best position that found by whole group (g_{best}):

$$V_{ik}(t+1) = \omega V_{ik}(t) + c_1 r_1 (p_{ik\,best}(t) - x_{ik}(t)) + c_2 r_2 (g_{k\,best}(t) - x_{ik}(t)) \quad . \tag{6}$$

$$x_{ik}(t+1) = x_{ik}(t) + V_{ik}(t+1) \quad . \tag{7}$$

Velocity update equation (6) will be simplified as (8) if consider the velocity and the best position found by each individual particle only:

$$V_{ik}(t+1) = \omega V_{ik}(t) + c_1 r_1 (p_{ik\ best}(t) - x_{ik}(t)) \quad . \tag{8}$$

Where ω is inertia weight; c_1 and c_2 are acceleration constants; r_1 and r_2 are random numbers between 0 and 1; index i indicates the *ith* particle; k $(1 \le k \le D)$ stands for the *kth* dimension of particle velocity vector and position vector. In the process of position update, particles cannot move out of the border and the velocity cannot exceed the limits i.e. $V_{ik}(t+1) \in [-V_{max}, V_{max}]$ and $x_{ik}(t+1) \in [-x_{max}, x_{max}]$. If any particle flies out of border, it will be redistributed in the area near the border:

$$\begin{cases} x_{ik}(t+1) = x_{\max} - \beta \cdot r() \cdot x_{\max} \\ x_{ik}(t+1) = -x_{\max} + \beta \cdot r() \cdot x_{\max} \end{cases}, \quad if \quad x_{ik}(t+1) \notin [-x_{\max}, x_{\max}] \quad . \tag{9}$$

where $\beta \in (0, 0.5)$; $r(\ )$ is random number between 0 and 1.

In order to make the particles search for optimum in a large range at beginning stage of iteration and converge quickly near the end of iteration, typically inertia weight ω decreases linearly from 0.9 to 0.4:

$$\omega(i) = 0.9 - \frac{i}{I_{\max}} \times 0.5 \quad . \tag{10}$$

where I_{max} is the maximum number of iterations.

2.2 The Guaranteed Convergence Particle Swarm Optimizer (GCPSO)

If follow standard PSO algorithm, a particle will stop nearby the current global best position if its velocity is near to zero, which implies all particles will stop there eventually if there is no new global best position is found. To correct this unwanted property, Van den Bergh introduced GCPSO algorithm and redefined the particle velocity and position update equations as below[3], [4]:

$$V_{\tau k}(t+1) = -x_{\tau k}(t) + p_{\tau k}(t) + \omega V_{\tau k}(t) + \rho(t)(1 - 2\,rand\ ()) \quad . \tag{11}$$

$$x_{\tau k}(t+1) = x_{\tau k}(t) + V_{\tau k}(t+1) = p_{\tau k}(t) + \omega V_{\tau k}(t) + \rho(t)(1 - 2\,rand\ ()) \quad . \tag{12}$$

where τ is the index of the current global best particle; *rand ()* is random number between 0 and 1. The value of $\rho(t)$ is adapted after each step according to (13):

$$\rho(t+1) = \begin{cases} 2\rho & if \quad \#\ successes > s_c \\ 0.5\rho & if \quad \#\ failures > f_c \\ \rho(t) & others \end{cases} \quad . \tag{13}$$

where *#successes* and *#failures* are the number of consecutive successes and failures respectively. The position update is defined as success if new position of a particle is different from that in last step; otherwise it will be defined as failure. This algorithm can achieve acceptable result when s_c=15, f_c=5 and $\rho(0)$ =1.

2.3 Niching Particle Swarm Optimizer

R. Brits *et al* introduced niching particle swarm optimization (NichePSO) algorithm to multiple optima detecting problem [2]. In NichePSO algorithm, if fitness change of a particle is less than δ over a certain consecutive number of iterations, then a subswarm is randomly created in the space around the particle with a radius of the Euclidean distance from the particle to its neighbor; subswarms are merged when they overlap or their Euclidean distance is less than μ. In the position update process, particles will be absorbed into a subswarm if they fly into a swarm. Normally δ and μ are small positive numbers. With NichePSO algorithm, both global optimum and local optima of multimodal functions can be detected. The flow and major steps of the NichePSO algorithm are as below [2]:

1. Initialize main particle swarm including particle quantity, particle position and particle velocity.
2. Train particles in main swarm per equation (7) and (8).
3. Update the fitness of particles in the main swarm and create subswarm if meet criteria.
4. For each subswarm:
    a. Train subswarm particles using one iteration of the GCPSO algorithm.
    b. Update each particle's fitness.
    c. Update swarm radius.
5. Merge subswarms if they meet merge criteria.
6. Allow subswarm to absorb those particles from the main swarm if they move into it.
7. Search in main particle swarm; create a new subswarm if any particle is found meeting partitioning criteria.
8. Repeat from 2 until stopping criteria are met.

3 NichePSO Algorithm Based Method for Process Window Selection

A proper fitness function is required for NichePSO algorithm. Fitness function below is constructed for parameters window selection problem:

$$Obj = |Q - R| \quad . \tag{14}$$

Apparently, *Obj* is small if *R* is close to *Q*, and it equals to 0 for the common points of *Q* and R. With NichePSO algorithm and (14), we developed a parameter window selection method, its flow and major steps for one dimensional problem is as below:

1. Search for common points of *Q* and *R* using NichePSO algorithm and fitness function (14). If none, return the information of no common point and exit program.
2. Sort border points together with the common points that found in step 1 in increasing order, pick a point x_j between each two neighbor points x_i and x_{i+1}; keep the portion $[x_i , x_{i+1}]$ if $Q(x_j)>R$.

3. Calculate the width $W_i = x_{i+1} - x_i$ of each kept portion, tag the portion as candidate interval if $W_i > 10.02\sigma$. If no candidate interval, return related information and exit program.
4. Output endpoints and the width of each candidate interval, and recommend the widest interval as the portion that parameter window should be selected on.

Two dimensional problems are more complex due to the solutions of $Q=R$ are curves when they intersect each other, which contain infinite common points. The flow and major steps for the two dimensional problems are as below:

1. Search for common points of Q and R using NichePSO algorithm and fitness function (14). If none, return information of no common point and exit program.
2. Calculate Euclidean distance between the common points that found in step 1. If the distance between a particle and its neighbor is less than ε, then search for common points again using NichePSO algorithm in the space around the particle with a radius of the Euclidean distance from it to its neighbor. Repeat this step until the distance between each particle and its neighbor is less than ε.
3. Link all common points that found in step 2 by following the rectangle grid model based algorithm for contour drawing [6].
4. Pick a point (x_i, y_j) in each area that separated by the close curves obtained in step 3, keep the area if $Q(x_i, y_j) > R$.
5. Calculate the widths of each kept area in both in x and y direction. If $W_x > 10.02\sigma_x$ and $W_y > 10.02\sigma_y$, tag the area as candidate area; if no candidate area, return related information and exit program.
6. Output the border lines and the widths W_x and W_y of each candidate area.

With geometric shape and widths of each candidate area, engineer can easily make judgment where the process parameter window should be selected from.

4 Simulation Results

R.Brits *et al* have ever used below standard multimodal functions to test NichePSO algorithm due to their typical properties of multimodal function [2], see figure 10. We also use the 5 functions to test the NichePSO algorithm based process parameter window selection method.

$$F1(x) = \sin^6(5\pi x) \quad . \tag{15}$$

$$F2(x) = (e^{-2\log^{(2)}\times(\frac{x-0.1}{0.8})^2})\times\sin^6(5\pi x) \quad . \tag{16}$$

$$F3(x) = \sin^6(5\pi(x^{3/4} - 0.05)) \quad . \tag{17}$$

$$F4(x) = (e^{-2\log^{(2)}\times(\frac{x-0.08}{0.854})^2})\times\sin^6(5\pi(x^{3/4} - 0.05)) \quad . \tag{18}$$

$$F5(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2 \quad . \tag{19}$$

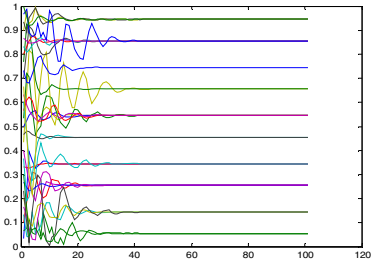


Fig. 3. Particle trace on F1

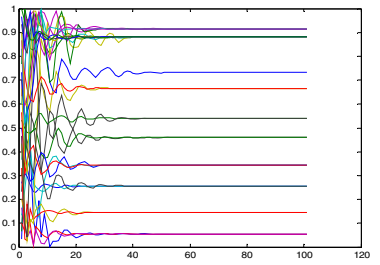


Fig. 4. Particle trace on F2

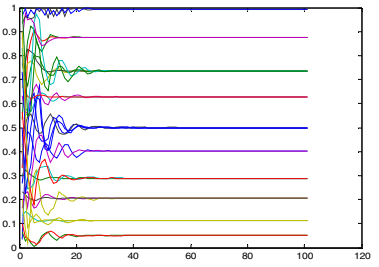


Fig. 5. Particle trace on F3

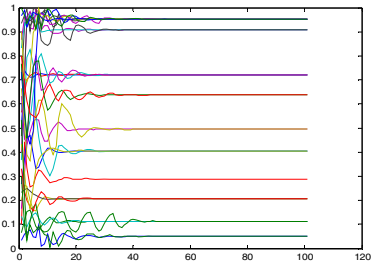


Fig. 6. Particle trace on F4

With main swarm particle quantity 30, $R=0.2$ and $\sigma=0.008$, we tested the method and got the results in table 1 and table 2. Table 1 shows all main particles converged at or close to the common points of R and each function, and all the common points in F1 to F4 were found during each test. Table 2 provides the recommended intervals which the parameters window can be selected on. Figure 3 to 6 record particle trace during iterations.

Table 1. Test summary

Function	R	σ	ε	Imax	Test Times	Fitness Mean	Fitness Deviation	%Common Points Found
F1	0.2	0.008	--	100	30	1.39E-05	5.31E-05	100%
F2	0.2	0.008	--	100	30	8.96E-07	2.47E-06	100%
F3	0.2	0.008	--	100	30	1.49E-04	4.72E-04	100%
F4	0.2	0.008	--	100	30	3.77E-04	1.34E-03	100%
F5	150	0.1, 0.1	0.3	500	30	8.01E-07	4.22E-06	--

Table 2. Test output (F1—F4)

Function	Endpoints of Candidate Interval					Width of Candidate Interval					Recommended Inerval
F1	0.0554, 0.1446	0.2554, 0.3446	0.4554, 0.5446	0.6554, 0.7446	0.8554, 0.9446	0.0892	0.0892	0.0892	0.0892	0.0892	1,2,3,4,5
F2	0.0555, 0.1445	0.2561, 0.3429				0.089	0.0868				1
F3	0.2057, 0.2894	0.4026, 0.5000	0.6280, 0.7359	0.8759, 0.9928		0.0837	0.0974	0.1079	0.1169		4
F4	0.2060, 0.2884	0.4054, 0.4950	0.6381, 0.7213			0.0824	0.0896	0.0832			2

F5 is a 3D curved surface with 4 maxima, which can be used to test two dimensional problems. Set main swarm particles as 300, with R=150, $\sigma_x=\sigma_y=0.1$ and $\varepsilon=0.3$, the test results on F5 are shown in table 3 and figure 9. Figure 8 is the image of F5 and R=150, and figure 7 records particles' original position, iteration-in-process position and final position respectively. Apparently, the particles' final positions in figure 7 match with the intersection curves of F5 and R=150 in figure 8. In most of real cases, the summary in table 3 and areas in figure 9 are good enough for engineer to define input parameter window.

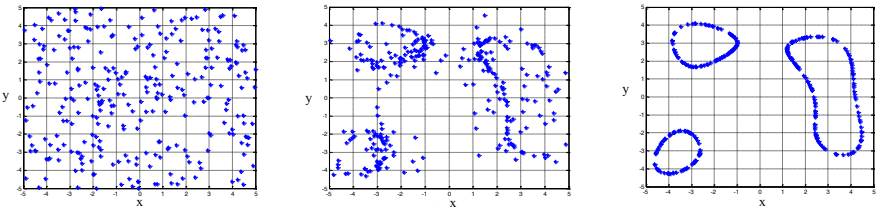


Fig. 7. Original, in-process and final position of particles (F5)

Table 3. Test output (F5)

Area	Wx	Wy
1	2.0091	2.3907
2	2.8521	2.4034
3	3.2886	6.5977

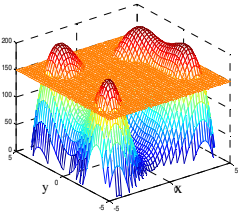


Fig. 8. F5 and R=150

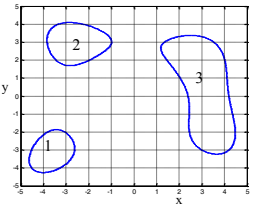


Fig. 9. Areas meeting requirements (F5)

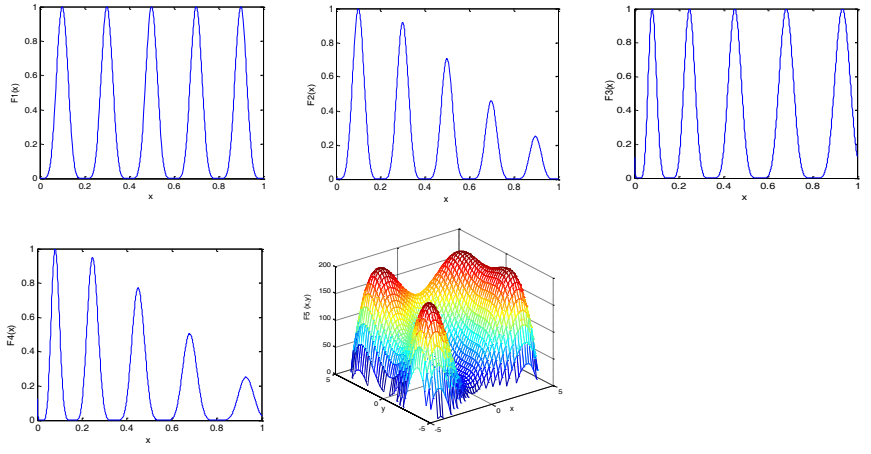


Fig. 10. Images of F1, F2, F3, F4 and F5

5 DOE Analysis and Production Validation

The distribution of output voltage V_{out} of a power management product is related to input voltage V_{in} and test temperature *Temp*. To achieve higher *Cpk* of V_{out} during test mass production, an experiment to define the ranges of V_{in} and *Temp* was conducted. Table 4 gives the matrix of the Design of Experiment (DOE) and results, where V_{out} mean and V_{out} stdev are average and standard deviation of V_{out} of 18 units tested in

same leg. Given the ideal target of V_{out} is $0.9V$ and preliminary specification limits are $0.888V$ and $0.912V$, *Cpk* in table 4 can be easily computed. The test head and resource channel of testing machine can provide stable V_{in} and *Temp* with standard deviations at $\sigma_{vin}=0.003V$ and $\sigma_{Temp}=0.6^oC$.

Set main particle quantity as 300 and take R=1.333, 1.5, 1.667 and 1.883, and then study the relation between V_{out} *Cpk* and input parameters V_{in} and *Temp* by using the method presented in section 3, then get the curves in figure 11. Engineer can easily select the area in rectangle as the process windows of V_{in} and *Temp* based on figure 11, i.e. $V_{in} \in$ [2.8V,5.2V] and Te*mp* $\in$ [0^oC, 60^oC], which can make V_{out} achieve higher *Cpk*. Set V_{in}=4.0V and *Temp*=30^oC as test mass production settings, then randomly pick 10 lots and record their V_{out} *Cpk* in table 5, here the distribution of V_{out} is much better than that we got during DOE, which shows the method presented in this paper is effective for process parameter window selection.

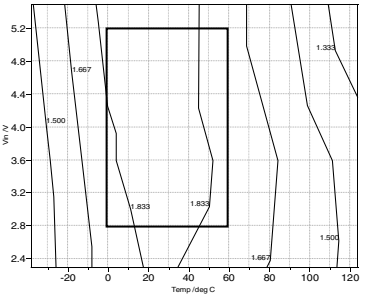


Fig. 11. DOE analysis result

Table 4. DOE matrix and data

Leg	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Temp /deg C	-40	-40	-40	0	0	0	25	25	25	85	85	85	125	125	125
Vin /V	3.6	2.3	5.5	3.6	2.3	5.5	3.6	2.3	5.5	3.6	2.3	5.5	3.6	2.3	5.5
Vout mean	0.89687	0.89688	0.89735	0.89876	0.89863	0.89923	0.89984	0.89956	0.90022	0.90055	0.90033	0.90145	0.90091	0.90043	0.90263
Vout stdev	0.00215	0.00217	0.00213	0.00199	0.00204	0.00198	0.002	0.00206	0.00198	0.00229	0.00237	0.00226	0.00261	0.00267	0.00261
Cpk	1.37847	1.36639	1.46676	1.80229	1.73727	1.89182	1.97496	1.87046	1.98523	1.66581	1.64371	1.55381	1.41683	1.44379	1.19414

Table 5. Production validation data

Lot	1	2	3	4	5	6	7	8	9	10
Cpk	1.98028	1.957399	1.98967	1.949872	2.01362	1.99061	1.99364	1.96887	2.01008	1.90897

6 Conclusion

This paper introduces a NichePSO algorithm based method for parameter window selection. For one dimensional problem, it can directly give the portion where input parameter window can be selected on; and it can provide the contours of output parameter and the width of each contour in both *x* and *y* direction for two dimensional problem. The production validation data and simulation results on standard multimodal functions reveal it is an effective method for parameter window selection.

Acknowledgement. This research is sponsored by National Science Foundation of China (70871091, 6107506 and 61034004), Ph.D. Programs Foundation of Ministry of Education of China (20100072110038), Foundation of the Ministry of Education of China for Returned Scholars and Program for New Century Excellent Talents in University of Ministry of Education of China.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, pp. 1942–1948 (1995)
2. Brits, R., Engelbrecht, A.P., Van den Bergh, F.: A Niching Particle Swarm Optimizer. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, pp. 692–696 (2002)
3. Wu, Q.D., Wang, L.: Intelligent Particle Swarm Optimization Algorithm Research and Application. Jiangsu Education Publishing House, Nanjing (2005)
4. Van den Bergh, F., Engelbrecht, A.P.: A New Locally Convergent Particle Swarm Optimizer. In: IEEE Conference on Systems, Man and Cybernetics, Hammamet, pp. 6–11 (2002)
5. Thomas, M.K., Donald, W.B.: The Certified Six Sigma Black Belt Handbook, 2nd edn. ASQ Quality Press, Milwaukee (2009)
6. Zhang, X.Q., Liu, Z.P.: An Algorithm of Contour Lines Based on Regular Grid. J. Computer Science. 32(9), 199–201 (2005)

Efficient WiFi-Based Indoor Localization Using Particle Swarm Optimization

Girma S. Tewolde and Jaerock Kwon

Department of Electrical and Computer Engineering
Kettering University, Flint, MI 48504, USA
{gtewolde,jkwon)@kettering.edu

Abstract. Location based services are rapidly gaining popularity in various mobile applications. Such services rely particularly on the capability to accurately determine the location of the user. Several techniques are already available to provide localization for static or mobile applications, GPS being the most popular. However, due to some limitations of GPS such as low accuracy, unavailability in indoor environments and lower signal quality in urban areas with high rise buildings, complementary solutions are essential to offer satisfactory service at all places all the time. This paper demonstrates the use of a widely available WiFi networking infrastructure for accurate and low-cost indoor localization. Most existing WiFi-based localization approaches employ radio signal strength indicator (RSSI) fingerprinting technique, which requires a great deal of pre-deployment effort. Our swarm-inspired optimization algorithm applies a simpler and efficient technique based on the radio propagation model of the wireless signal. The proposed technique is evaluated in simulation and is demonstrated to achieve excellent average localization error of about 4 meters in an area of 50 x 50 square meters, under noisy RSSI measurements.

Keywords: indoor localization, RSSI modeling, RSSI fingerprinting, particle swarm optimization.

1 Introduction

The recent advances in mobile Internet technology and the proliferation of a wide range of services are further accelerating the research and development effort for improved as well as new applications. Modern location based services (LBS) have evolved as a result of the technology integration of mobile Internet, smart phones, and Geospatial information. Knowledge of the user's location as well as locations of interest points in a given area are at the core of LBS and related services.

The problem of localization has been a core focus of research by many researchers in various fields, including mobile robotics, wireless sensor networks, mobile communication and internet technology. In general, different techniques may be utilized for outdoor and indoor localization. The focus of our study in this paper is indoor localization. Simplicity, cost, and accuracy of localization are common criteria for comparing different localization strategies. Ideally, it is desirable to have a

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 203–211, 2011.
© Springer-Verlag Berlin Heidelberg 2011

technique that could be easily deployable without requiring initial setup, training or environmental adaptation. It is also necessary that the technique provides sufficient accuracy for the class of applications it is intended for. Obviously, a low cost solution is desirable to make the technology affordable for widespread applications.

Several techniques such as LANDMARC [1], Cricket [2], and Active Badge [3] are already available for indoor localization. However, these techniques deploy and rely on specialized infrastructure based on RFID, sonar, IR, or radio signals for the purpose of the localization thus adding to the deployment cost. Although the initial setup adds up to the cost of localization, these techniques often provide excellent accuracies. For example, Cricket - a triangulation based positioning technique using active ultrasound beacons, achieves an overall localization accuracy within 10 cm of the actual position. Such high levels of accuracies could be attractive for applications such as mobile robot navigation and object manipulation tasks.

On the other hand, several indoor and outdoor applications may not necessarily require such high levels of sub-meter accuracies. For example, when a person is navigating in an unknown indoor office environment or outdoors it might be sufficient to be able to localize within a few meters to aid find nearest interest points. For such applications where the accuracy requirement is relatively relaxed simpler and lower-cost localization techniques will be more attractive.

Two related radio frequency (RF) based localization techniques attract great interest mostly due to the fact that existing wireless communication infrastructure is exploited without incurring additional deployment cost. These methods rely on RF signals either from cellular towers or WiFi access points (APs). The localization solution based on reference information from cellular towers can be utilized anywhere cellular signals from at least 3 towers are available. For example, as part of the Federal Communications Commission (FCC) E911 rules wireless carriers in the US are required by law to provide a 911 caller's location to within 50 to 300 meters depending on the technology used [4]. Despite the limited accuracy of this technique it applies both indoors and outdoors providing global localization in extended geographical areas.

Our proposed solution is based on RF signals from WiFi access points. It assumes knowledge of the locations and transmit-powers of at least three access points. To minimize the effect of RSSI measurement uncertainties in the localization accuracy we employ a particle swarm based optimization algorithm that is found to result in robust performance in noisy environments. The main contribution of this paper is that we demonstrate an efficient and sufficiently accurate localization scheme for indoor localization using existing infrastructure with little initial setup or training.

The rest of this paper is organized as follows. Section II presents related work on RF based localization. Section III presents the wireless signal propagation model that is used in our localization solution. Section IV gives a short introduction to the PSO algorithm and describes the mapping of the localization problem to the PSO model. Section V presents the experiment and results. And finally the paper provides discussion and concluding remarks in sections VI and VII.

2 Related Work

In general, the localization techniques that use RF signals on existing networks employ a radio signal fingerprinting approach, model based techniques, or a combination with other localization methods [5]. In the radio signal fingerprinting method the RSSI measurements from all the radio transmitters in the region is mapped at each location of the environment. To localize a given device its RSSI readings are matched against the values stored in the map. RADAR [6] is an example of this technique which uses extensive offline data collection before real-time localization starts. There have been several improvements on this approach through the years as well as commercialization of the technology [7].

In a similar approach to RF signal fingerprinting but at a global scale a company called Skyhook [8] developed a WiFi positioning system (WPS). It collects and maintains a massive worldwide database of WiFi access points in major populated areas. Using the data and applying intelligent search techniques the company provides subscribers real-time access to location information. The company claims an accuracy of 10 to 20 meters by its core engine.

Alternative to the signal-fingerprinting and map-based approaches is a model-based technique that relies on the radio propagation property of the WiFi signal. In this case, the RF propagation model is used to predict the RSSI at various points in an environment. This method eliminates the cost of initial deployment, maintenance and the issue of scalability associated with the signal fingerprinting technique. But its localization accuracy may be slightly lower [9]. Previously proposed solutions using this approach include Chintalapudi et al. [5], Lim et al. [10], Madigan et al. [11]. Most of these techniques except [5] assume knowledge of the locations and transmit power of the APs, and/or rely on WiFi sniffers at known locations to provide anchor points for the localization algorithms. Chintalapudi et al. on the other hand assume access to GPS data at some locations in the environment. The GPS data provides known location fixes for the environment modeling created by a server. The system is configured in client-server model so mobile nodes query the server for their location information by sending requests wirelessly.

Like many of the existing techniques, our proposed method assumes knowledge of the locations and transmit characteristics of at least three access points in the operating environment. We modeled the localization problem as an optimization problem with the goal of minimizing the computed location error. Then an intelligent nature inspired problem solving strategy is applied to the optimal solutions for the localization problem. In this paper we employed the Particle Swarm Optimization (PSO) algorithm. The main reason for choosing PSO over other competing optimization techniques is due to its simplicity and proven performance to deal with noisy optimization problems. For example, on a similar problem of emission source localization in noisy environments we demonstrated the superior performance of PSO over Differential evolution (DE) and Matlab's non-linear least squares (LSQ) optimization tool [12].

3 Wireless Signal Propagation Model

In our studies we assume an environment with IEEE 802.11 wireless communication at 2.4 GHz band. There are several experimental and theoretical studies of radio signal propagation in indoor environments [13]. In this paper the log-distance path loss (LDPL) model is used to predict RF signal attenuation as a function of distance between an AP and a WiFi receiver. This model is given by Equation (1) below:

$$p_d = P_0 - 10 \cdot \alpha \cdot \log(d) + R. \tag{1}$$

where pd is the received power in dBm at distance d (in meters) from the transmitter. P0 is the signal strength 1 meter from the transmitter, α is known as the path loss exponent, and R represents a random variable for capturing the variations in the RSSI readings due to multi-path effects, physical barriers in signal path and other imperfections in the model. The parameter α is dependent on the environment, i.e. type of construction material, architecture, location, temperature, humidity, etc. Empirical measurements of α in the literature report values in the range 1.8 to 5 depending on the level of obstruction [11]. Lower value of α correspond to lower signal path loss. For example, for free space propagation a value of 2.0 is used and for office environment with wall partitions, furniture and people a value of 2.5 would be a reasonable choice.

4 Particle Swarm Based Localization Algorithm

Particle Swarm Optimization (PSO) was inspired by the social swarming behavior of bird flocks, fish schools, and bee swarms. It was first developed in 1995 by Kennedy and Eberhart [14]. Individual particles in a particle swarm represent candidate solutions for the optimization problem. Initially, at the start of the optimization algorithm the PSO particles are assigned random initial positions in the search space. The particles are then moved around in the parameter space by using systematic rules to adjust their velocities and positions, in response to the swarm's experience in locating quality solutions.

The social interaction of the particles in the swarm shapes the dynamics the optimization algorithm. Thus, the performance of the individual particles in the swarm is influenced by a combination of their personal and social best experiences. In effect, the particles tend to be attracted to the best solution they have individually found and the best solution that any particle in their neighborhood has found.

Different PSO models have been proposed over the years to target different classes of problems. Two of the most widely known models are the constriction-factor and inertia-weight forms of the PSO algorithm, both of which have been demonstrated to be effective for general optimization tasks. The governing equations for the constriction-factor form of the PSO algorithm are given by Equations (2) and (3):

$$v_{ij}(t+1) = \chi \cdot \left(v_{ij}(t) + \varphi_1 \cdot r_1 \cdot (p_{ij} - x_{ij}(t)) + \varphi_2 \cdot r_2 \cdot (p_{gj} - x_{ij}(t)) \right). \tag{2}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{3}$$

The quantity χ is called constriction-factor. The quantity is the personal best position of particle and is the global best position in the entire swarm. and represent the learning rates that control the degree of influence of the cognitive and social components. and are independently generated random numbers in (0,1). They contribute to the stochastic behavior of the algorithm to allow random exploration of the search space in the surroundings of the personal and neighborhood best positions. The performance of each particle is measured using a problem specific pre-defined fitness function.

In our proposed solution strategy for the problem of localization we assume the presence of at least three reference APs with known locations and transmit powers. When we want to compute the (x, y) location of a point in space within the operating environment, we will first collect RSSI readings from the three APs. Then we apply the swarm inspired optimization method on the LDPL model to reach at an optimal localization with minimum estimation error. The fitness function for the optimization problem is derived with the goal of minimizing the sum of the squares of the errors between the actual RSSI readings from all the APs at the unknown location (x, y) and the theoretical values that would be obtained from the LDPL model, computed over all the reference APs.

$$f = \sum_{\forall APs} \left(p_i - P_{0i} + 10 \cdot \alpha \cdot \log(d_i)\right)^2. \tag{4}$$

where, pi is the RSSI reading at the unknown location from the ith access point. P0i is the RSSI reading at 1 meter radius from the ith AP. di is the distance in meters of the unknown node location from the ith AP. In equation (4) di is the unknown quantity that will be solved as part of finding the solution for the unknown location (x, y).

The PSO algorithm initially starts by assigning the particles in the swarm random positions in the search space. By evaluating the fitness function for each particle the algorithm determines how close they are to the actual position. Particles that are located far away from the desired location (x, y), whose position is being computed, result in larger values of f corresponding to higher estimation errors than particles closer to the actual node location. Then, in successive iterations the algorithm tries to update the particles' velocities and positions using the PSO equations so as to improve their fitness values.

5 Experiment and Results

For our simulation experiment we considered the following environmental layout, shown in Fig. 1. Three different dimensions of the same environmental layout were considered to study the effect of the size of the working areas on the estimation accuracy. Experiments were conducted on the following three sizes of the environment:

a) 25 x 25 square meters
b) 50 x 50 square meters
c) 100 x 100 square meters

In each case three APs were placed far apart from each other in a triangular formation. Then randomly generated points were distributed all over the environment. The PSO based localization algorithm computes the locations of these points based on their RSSI measurements.

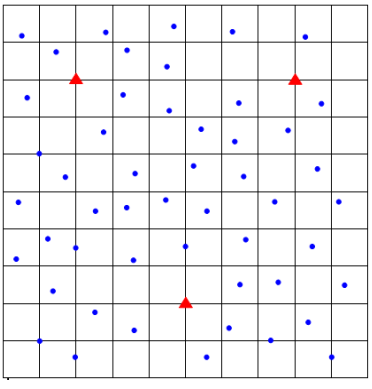


Fig. 1. Layout of the test environment with randomly placed test points (the triangles are APs, the dots are points whose localization is to be computed)

The only information the optimization algorithm has at the its start is the set of RSSI measurements at the unknown position from the three WiFi access points. The locations and transmit powers of the access points is assumed known, and also the path loss model of the radio signal in the operating environment.

To study the impact of the measurement uncertainties in the RSSI values due to environmental effects, zero mean Gaussian distributed random noise was introduced in the emulated readings of the radio signals. The standard deviation of the noise was varied from 0 to 5, to see how it impacts the localization accuracy.

To minimize the impact of noise the localization algorithm takes 30 samples of the RSSI readings at each unknown location and takes the average value. Using the RSSI readings and knowledge of information about the three reference access points the PSO algorithm computes the localization of each point. For the LDPL model in the problem formulation we used a value of $\alpha = 2.5$ which was assumed to work fine for a single floor office building with some partitioning walls and corridors.

The PSO parameters used in the simulation are swarm size of N = 50, constriction factor, $\chi = 0.729$, learning rates, = = 2.05.

The performance of the localization algorithm is evaluated by calculating the localization error which is computed as the Euclidean distance between the estimated location (x', y') of the point from the algorithm and its actual location (x, y).

Fig. 2, 3 and 4 present the way the localization algorithm performs over the three environmental dimensions.

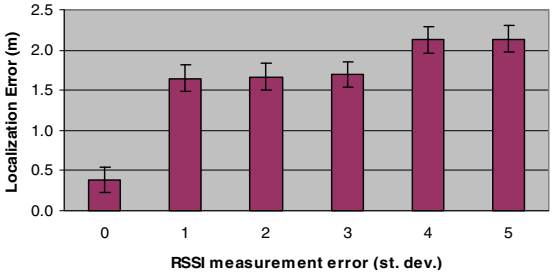


Fig. 2. Localization accuracy on 25 x 25 sq. meters. The horizontal axis shows st. dev. of the Gaussian noise in the RSSI measurement error.

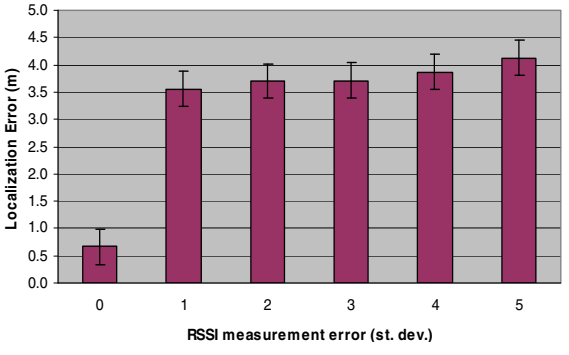


Fig. 3. Localization accuracy on 50 x 50 sq. meters. The horizontal axis shows st. dev. of the Gaussian noise in the RSSI measurement.

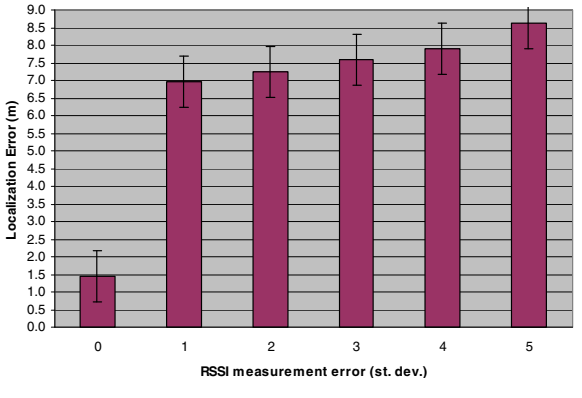


Fig. 4. Localization accuracy on 100 x 100 sq. meters. The horizontal axis shows st. dev. of the Gaussian noise in the RSSI measurement.

6 Discussion

From Fig. 2, we see that for the smallest size operating environment (25 x 25 sq. meters), the average maximum localization error is limited to about 2 meters for the maximum setting of the noise level. From Fig. 3, for the 50 x 50 sq. meters operating environment, the average maximum error is shown to be limited to about 4 meters for the maximum noise level. And from Fig. 4, for the 100 x 100 sq. meters operating environment, the average maximum error is shown to be about 8.5 meters at the maximum noise level. From these results we observe that the localization error follows more like a linear pattern as a function of the dimension of the operating environment.

To compare the performance of our algorithm to that of RSSI fingerprinting techniques, consider the problem dimension 50 x 50 square meters which is close to the size of the environment reported in RADAR [6]. The 50 percentile (median) error of our algorithm at the maximum level of Gaussian noise is about 3.4 meters. This result is only slightly worse than the best 2 to 3 meter achieved by RADAR which relies on cumbersome offline data collection.

A resolution of 3 to 5 meters for medium size environments could be quite good enough for common indoor applications. Our goal is to deploy such localization tools on autonomous mobile robots that could benefit from the WiFi based global localization for navigational tasks, while improvements on the accuracy can be achieved using additional sensors for local perception. For example, if vision or other sensors are available on board the mobile robot, information obtained from visual landmarks could be used to aid in improving the localization accuracy.

When examining the execution performance of our proposed algorithm, we find that compared to other model based methods the PSO-based solution is found to be efficient and converges fast in less than one second on a Dell Latitude E5400 laptop. In contrast, the result reported in [8] takes extra long off-line training times (16 to 65 minutes on a Lenovo T61p laptop) for building the RF model using Genetic Algorithm based technique.

7 Conclusion

The significance of our PSO based localization technique using existing WiFi infrastructure is that it requires little initial setup. It can be easily deployed for use by humans or mobile robots as long as WiFi access from at least three APs can be obtained. Unlike other techniques, such as [5] and [8], there is no offline training required. These are important properties especially in scenarios when there is no prior information about the environment or when there is no time to gather fingerprinting data that would be needed for some of the other methods.

The proposed technique is evaluated in simulation and is demonstrated to achieve average localization error of about 4 meters in an area of 50 x 50 square meters, under noisy RSSI measurements. This is reasonably sufficient accuracy for the class of target applications the technique is meant for. In future work we will implement and carry out experiments in real physical environments and evaluate its performance. Methods of automatically determining the parameters of the APs will also be examined.

References

1. Ni, L., Liu, Y., Yiu, C., Patil, A.: LANDMARC: Indoor Location Sensing Using Active RFID. In: WINET (2004)
2. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket Location-Support System. In: MobiCom (2000)
3. Want, R., et al.: The Active Badge Location System. ACM Transactions on Information Systems (January 1992)
4. http://www.fcc.gov/cgb/consumerfacts/wireless911srvc.html
5. Chintalapudi, K., Iyer, A.P., Padmanabhan, V.N.: Indoor Localization without the Pain. In: MobiCom (2010)
6. Bahl, P., Padmanabhan, V.N.: RADAR: An In-building RF-based User Location and Tracking System. In: INFOCOM (2000)
7. Ekahau, http://www.ekahau.com/
8. Skyhook, http://www.skyhookwireless.com/
9. Gwon, Y., Jain, R.: Error Characteristics and Calibration-free Techniques for Wireless LAN-based Location Estimation. In: MobiWac (2004)
10. Lim, H., et al.: Zero Configuration Robust Indoor Localization: Theory and Experimentation. In: Infocom (2006)
11. Madigan, D., et al.: Bayesian Indoor Positioning Systems. In: Infocom (2005)
12. Tewolde, G.S., Hanna, D.M., Haskell, R.E.: Particle Swarm Optimization for Emission Source Localization in Sensor Networks. In: The 2009 Artificial Neural Networks in Engineering (ANNIE 2009) Conference, St. Louis, MO (2009)
13. Rappaport, T.S.: Wireless Communications - Principles and Practice. Prentice Hall PTR, Englewood Cliffs (1996)
14. Eberhart, R., Kennedy, J.: A new Optimizer using Particle Swarm Theory. In: 6th Int. Symp. Micro Machine and Human Science (1995)

Using PSO Algorithm for Simple LSB Substitution Based Steganography Scheme in DCT Transformation Domain

Feno Heriniaina Rabevohitra and Jun Sang*

School of Software Engineering, Chongqing University, Chongqing, China 400030
fenoheriniaina@gmail.com, jsang@cqu.edu.cn

Abstract. An improved method for embedding a secret message into a cover image with least significant bit (LSB) substitution in discrete cosine transformation (DCT) domain was proposed. The secret message was first split into partitions, while the cover image was divided into blocks of size 2x2, and DCT was used to convert the blocks from spatial domain to frequency domain. Then, Particle Swarm Optimization (PSO) algorithm was applied to search for an optimal substitution matrix T to transform the split partitions for an optimal embedding. Next, the transformed part of secret message was embedded into the AC coefficients of the transformed image blocks by LSB substitution. Experimental results show the proposed method can keep the quality of the stego-image better, while the security of the hidden secret message is increased by use of the substitution matrix T.

Keywords: Steganography, PSO, LSB, DCT.

1 Introduction

Internet is the world on top of our fingers and it is anything but secure. Information can be sent from the half part of the globe to the other half in just seconds. However, sharing data over public networks such as the Internet is unsafe. Ways to insure data confidentiality is then needed. In contrast with cryptography, steganography is the art and science of hiding a message in a cover signal for transmission where the concealed message will not be apparent to anyone except for the targeted receiver to be able to extract its content with help of pre-defined way or key.

The most well-known steganographic technique in the data hiding field is LSBs substitution [1]. Image hiding methods can generally be categorized into spatial and frequency domain [2]. In 2007, a JPEG steganographic method was presented by Li and Wang [2]; they have modified the quantization table used in JQTM (Jpeg and quantization table modification) for embedding larger secret message into the DC-to-middle frequency components for each block and developed a PSO algorithm to approach optimal LSB substitution, which proposed a higher security level and a better quality of the stego image. Recently, a method proposed in [3] imbedded data at the LSBs of 2x2 transformed blocks of the cover image in frequency domain. Based on LSB embedding in frequency domain, using PSO and a key transformation

* Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 212–220, 2011.
© Springer-Verlag Berlin Heidelberg 2011

for embedding a data and recovering the hidden data is presented in this paper. Implementations and experiments show good PSNR values.

The rest of this paper is organized as follows: Related works are introduced in Section 2. The proposed schemes are presented in section 3. Experimental results are illustrated in section 4 and section 5 is for comments and conclusion.

2 Related Works

2.1 DCTIASMTT

DCTIASMTT [3] presents a method for embedding secret data into gray scale image file which is split into blocks of 2x2 transformed in frequency domain using DCT and then embedded in its LSB pixel values (excluding the first pixel). Inverse DCT is then used to get the image back to spatial domain.

Presented below an illustrative figure which shows the encoding scheme of the DCTIASMTT and the decoding is done using reverse procedure.



Fig. 1. Encoding scheme using DCTIASMTT

2.2 Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is a technique used to explore the search space of a given problem to find the settings or parameters required to maximize a particular objective [4]. This technique, first described by James Kennedy and Russell C. Eberhart in 1995 [5], originates from two separate concepts: the idea of Swarm intelligence based on the observation of swarming habits by certain kinds of animals and the field of evolutionary computation. A PSO algorithm works by simultaneously maintaining several candidate solutions in the search space. During each iteration of the algorithm, each candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be thought of as a particle "flying" through the fitness landscape finding the maximum or minimum of the objective function [4]. It is the objective function which is used to evaluate candidate solutions and operates on the resultant fitness values of the PSO algorithm.

2.3 A Steganographic Method Based Upon JPEG and Particle Swarm Optimization Algorithm

Published in 2007, Xiao Li and J. Wang [2] presented a steganographic method based on JPEG compression and PSO. PSO was used for finding an optimal substitution matrix for transforming the secret message before its embedding in the DC-to-middle frequency components of the quantized DCT coefficients of the cover-image during

the jpeg compression phase. This resulted of a higher security level of the stego image compared to JPEG and Quantization Table Modification (JQTM) [6].

3 Proposed Schemes

3.1 Three Different Schemes

Here, we present 3 different schemes for improving the method proposed in DCTIASMTT. All based on the use of table matrix transformation selected by PSO before embedding, the size of the table matrix, the processing time and the quality of the stego image resulting from the use of each of them are different. For our work we use gray scale image of size (MxN). The technique inserts a secret message of maximum-size (M/2xN/2x3)-16bits. DCT is used to transform the image from spatial domain to frequency domain. The secret data gets imbedded in the frequency domain of the cover image.

Scheme 1: Using a unique table transformation to transform the whole block of the secret data.

Encoding process is as follows:

1. Load the source image and secret data.
2. Convert the secret data into stream of bit stream.
3. Convert the cover image into blocks.
4. By use of DCT, transform each block of the cover image into its discrete cosine transform values.
5. Retrieve the k-LSBs of each of the cover image block
6. Search for Optimal table transformation matrix T by use of PSO
    a. Get a table transformation matrix T* by use of PSO
    b. Convert the binary number of the secret message part to decimal and transform the secret data using T* before getting it back to binary values
    c. Calculate the PSNR for each substitution of the secret data with the retrieved k-LSBs of the cover image
    d. Select the one having the highest PSNR as an optimal transformation of the secret data and the table transformation used as best transformation matrix T.
7. Embedding followed by inverse discrete cosine transform
8. Save table transformation T and stego image

Using this system, only one transformation matrix is used to map each block of k bits of the secret data into new block of k bits. The size of T is kept 2^k

Scheme 2: Using a unique table transformation to transform each block of the secret data and save all used tables to compose T.

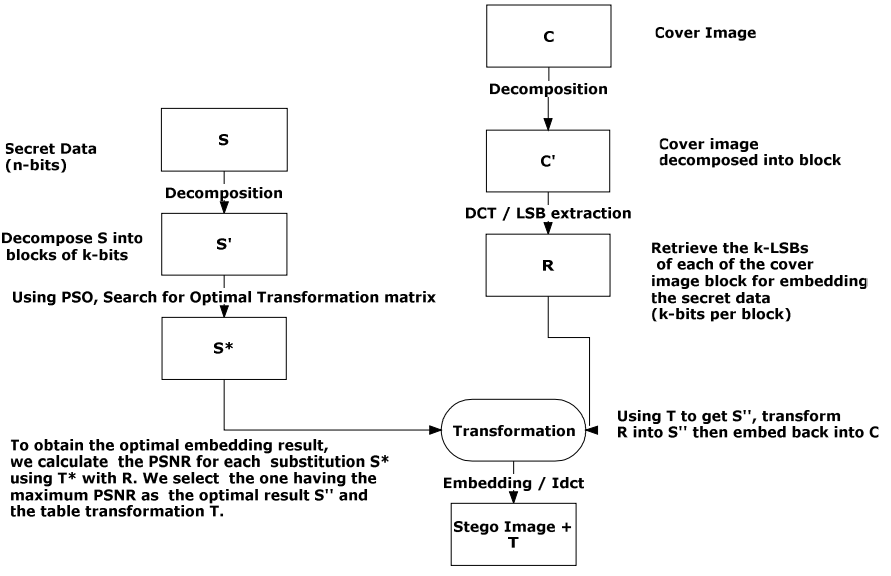


Fig. 2. Encoding flowchart by use of PSO, transformation domain and LSB substitution, scheme 1. Decoding requires the possession of T.

Figures 3 and 4 below show the encoding and decoding scheme of the proposed method:

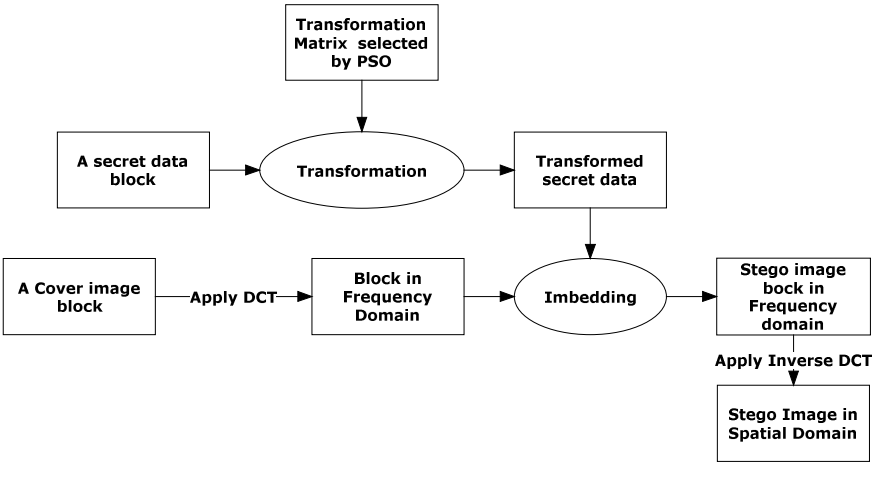


Fig. 3. Encoding flowchart using PSO, transformation domain and LSB substitution, scheme 2

Decoding requires the possession of all tabular matrices previously used to transform each block of the secret data presented as T in figure 4.

To reconstruct the secret message, inverse transformations of the encoding in the opposite order is used with the saved table transformation T and stego image as inputs.:

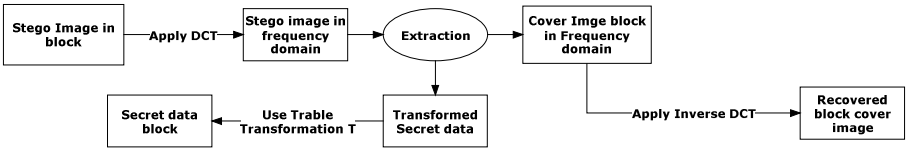


Fig. 4. Decoding flowchart, scheme 2

Scheme 3: it is only different from scheme two for the setting of PSO`s objective function which we will describe in the next section.

3.2 PSO Settings

The information is stored in the frequency domain, DCT coefficients of the cover image. We assume the cover image is split into 2x2 blocks and that every block encodes 3 bits. We build the PSO algorithm to select an optimal table transformation matrix T as in [2] section 2.2 where in our case k is equal to 3.

Objective functions:

- Scheme 1: The objective function minimizes the error on the stego image. We measure the cost of transformation between original discrete cosine transform value of the cover image and the transformed value of the secret message. This will now be referred to PSO1
- Scheme 2: using PSO 2 set as to minimize the error on the stego image. The average intensity distortion between the cover image 2x2 blocks and the 2x2 Stego blocks produced is used as objective function.
- Scheme 3: uses PSO 3 and the average intensity distortion between the original secret data and the recovered secret data is the objective function. It is defined to minimize the error on the hidden data.

Every member of the population is an 8 dimensional vector. And sorting this vector is the transformation of the member. Each particle represents a transformation. The best particle attracts the others and the best one of the last generation gets returned as the solution to the problem.

Each block to be embedded from the cover image will be converted from spatial domain to frequency domain using DCT. We replace each bit of the DC transform value for all blocks where the value is not equal to the considered bit of the transformed secret message. For any of the modified discrete cosine transform values that have changed, apply inverse discrete cosine transform to get back the image block in spatial domain.

4 Experimental Results and Discussions

To evaluate the performance of the proposed methods, we implemented the scheme described in "Discrete Cosine Transformation Based Image Authentication and Secret Message Transmission scheme" and our proposed schemes. We develop 3 PSO algorithms which differ for the calculation of the cost of transformation: PSO1, PSO2 and PSO3 as presented in 3.2.

All these programs were coded in Matlab 7.10 and run on a PC with Intel Core i5 CPU M460 2.53GHz; with 4GB RAM under Windows 7-64 operating system. And as we focus on the stego image quality with the maximum size of the hidden data limited to (M/2xN/2x3) -16bits, our criteria for evaluations are PSNR and MSE.

We use the 8-bit gray-level images "Baboon", "Boat", "Lena" and "Peppers" as cover-images, and for all our experiments "Plane" will be our hidden data.



Fig. 5. Our cover images (Baboon, Boat, Lena and Peppers) and the secret data (Plane of size 75x75)

The following tables relate the results of our experiments:

Table 1. Comparing PSNR values between Source image and stego image, presented on this table are the results from DCTIASMTT, PSO1 (our proposed method scheme 1), PSO2 (our proposed method scheme 2 where objective function focuses on minimizing the error on the stego image) and PSO3 (our proposed method scheme 3 and objective function gives focus on minimizing the error on the secret message)

		PSNR value in DB			
		Method			
Images	*Image Size*	DCTIASMTT	PSO1	PSO2	PSO3
Baboon	512*512	51.1496	51.2214	57.1888	51.1798
Lena	256*256	46.5180	46.5446	51.4310	46.5400
Boat	256*256	46.2311	46.2034	51.4408	46.2157
Boat	512*512	52.6235	52.6488	57.4641	52.6235
Peppers	256*256	46.4566	46.4839	51.4452	46.4709

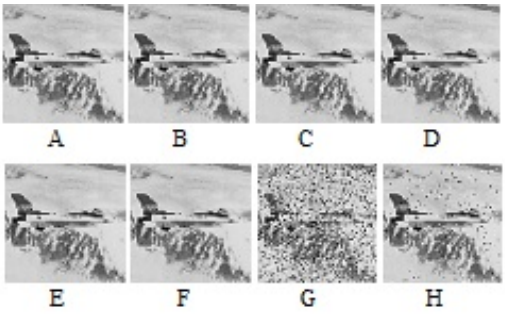


Fig. 6. shows "Plane.bmp" (75x75) images used as secret data in all our experiments. From "B" to "H" were all extracted from Baboon after all the different methods of imbedding that are listed below. Image "A" is the original secret data.

Table 2. Secret data resulting from all our proposed schemes

Image	*Method used for embedding*	*Image block*
A	Plane (our secret data)	
B	DCTIASMTT	2x2
C	Our proposed method PSO1	2x2
D	Our proposed method PSO2	2x2
E	Our proposed method PSO3	2x2
F	Our proposed method PSO1	4x4
G	Our proposed method PSO2	4x4
H	Our proposed method PSO3	4x4

During our experiments, we have also implemented the same proposed scheme 2 and scheme 3 in difference of the block Image size of 2x2 to be 4x4

Table 3. Experimental results of the proposed method scheme 2 and 3, in the difference of the 2x2 image block replaced to be 4x4.

		PSNR(dB)	
Images	Image Size	PSO2	PSO3
Baboon	512x512	53.3389	51.0174
Lena	256x256	48.2312	46.3531
Boat	256x256	47.9865	46.0007
Boat	512x512	54.2615	52.5407
Peppers	256x256	48.1262	46.2300

Table 4. Processing time of DCTIASMTT and the proposed methods for 2*2 image block size

		Processing Time in Seconds			
		Method			
Images	*Image Size*	DCTIASMTT	PSO1	PSO2	PSO3
Baboon	512*512	31.665	299.792	355.264	467.540
Lena	256*256	14.181	276.310	364.444	469.060
Boat	256*256	14.195	284.677	364.343	461.239
Boat	512*512	31.628	307.950	367.709	467.221
Peppers	256*256	14.198	272.339	363.911	465.707

5 Conclusion

This paper proposed a method for concealing information with LSB substitution in frequency domain while using PSO to optimize the embedding procedure.

Compared with DCTIASMTT in [3], by use of PSO for finding an optimal transformation matrix T, the proposed method results a minute increase in efficiency, while increasing the security level.

And compared with the method in [2], the proposed method is much simpler, since it only uses DCT, not the whole procedure for JPEG compression. For scheme 2 and 3, the table transformation for each block can be searched using PSO although that's not optimal at all in the setting using 2x2 blocks from the source image. The best transformation is the one where the part of the secret message gets mapped to the exact bit representation of the considered bits of the source image. And the stego-image will be exactly the same as the source image as no bit needs to be altered. As for 2x2, exact calculation can generate the best transformation. We implemented 4by4 to execute the true potential of PSO. Here, schemes 2 and 3 offer a highly improved stego image quality with cumulated tables to form T needed for the re-transformation. And scheme 1 provides a higher security of the stego image compared to DCTIASMTT with an acceptable PSNR value and smaller T size.

Acknowledgments. This work was supported by National Natural Science Foundation of China (No. 60972105) and Natural Science Foundation Project of CQ CSTC (No.2009BB2210).

References

1. Bender, W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for data hiding. IBM Systems Journal 35(3-4), 313–336 (1996)
2. Li, X., Wang, J.: A steganographic method based upon JPEG and particle swarm optimization algorithm. Information Sciences 177, 3099–3109 (2007)

3. Bhattacharyya, D., Dutta, J., Das, P., Banyopadhyay, S.K., Kim, T.-h.: Discrete cosine Transformation based image authentication and secret message transmission scheme. In: First International Conference on Computational Intelligence, Communication Systems and Networks (2009)
4. Blondin, J.: Particle Swarm Optimization: A tutorial (September 4, 2009)
5. Kennedy, Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001) ISBN: 1-55860-595-9
6. Chang, C.C., Chen, T.S., Chung, L.Z.: A steganographic method based upon JPEG and quantization table modification. Information Science 141, 123–138 (2002)
7. Parsopoulos, K.E., Vrahatis, M.N.: Particle Swarm Optimization and Intelligence: Advance and Applications. Information Science (2010)
8. Fazli, S., Kiamini, M.: A High Performance Steganographic Method Using Jpeg and PSO Algorithm. In: IEEE International Multitopic (December 24, 2008)
9. Fridrich, J.: Steganography in Digital Media Principles, Algorithm, and Applications. Cambridge University press, Cambridge (2010)

Numerical Integration Method Based on Particle Swarm Optimization

Leila Djerou[1], Naceur Khelil[2], and Mohamed Batouche[3]

[1] LESIA Laboratory, Mohamed Khider University at Biskra, Algeria
ldjerou@yahoo.fr
[2] Laboratory of applied Mathematics, Mohamed Khider University at Biskra, Algeria
khelilna@yahoo.fr
[3] Computer Science Departement CCIS-King Saud University, Riyadh Saudi Arabia
mbatouche@ccis.ksu.edu.sa

Abstract. In this paper, a novel numerical integration method based on Particle Swarm Optimization (PSO) was presented. PSO is a technique based on the cooperation between particles. The exchange of information between these particles allows to resolve difficult problems. This approach is carefully handled and tested with some numerical examples.

Keywords: Numerical integration, Particle Swarm optimization, trapezoidal rule.

1 Introduction

Solving numerical integration is an important question in scientific calculations and engineering. As is well known, that for a partition X[1] of $[a, b]$ and a function $f : x \in [a, b] \longrightarrow R$ bounded on the interval $[a, b]$, is integrable, if and only if, $\forall \varepsilon > 0, \exists X$ (partition) over $[a, b]$ s.t.$\sum_{i=0}^{N}(L_i - l_i)(x_i - x_{i-1}) < \varepsilon$, where $L_i = sup_{[x_{i-1}x_i]}f(x)$ and $l_i = inf_{[x_{i-1}x_i]}f(x)$. Therefore, the purpose of this problem is to find a partition X of $[a, b]$ that returns (in a certain manner) the quantity $\sum_{i=0}^{N}(L_i - l_i)(x_i - x_{i-1})$ the smallest possible. The quality of the result depend on the choice of X. How to choose the best partition? To do this, conceptually, we would like to select the partition using particle swarm optimization.

This paper is organized as follows. In Section 2, a formulation adapted to the strategy of particle swarm optimization and the construction of an algorithm to generate the different agents in a swarm. The Section 3 exposes some essential examples to show how the PSO algorithm can lead to a satisfactory result for numerical integration. The comments and conclusion are made in Section 4.

[1] A partition X is a finite set of points : $a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b$.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 221–226, 2011.
© Springer-Verlag Berlin Heidelberg 2011

2 Overall Description Strategy of Particle Swarm Optimization

Recently, a new stochastic algorithm has appeared, namely 'particle swarm optimization' PSO. The term 'particle' means any natural agent that describes the 'swarm' behavior. The PSO model is a particle simulation concept, and was first proposed by Eberhart and Kennedy [2]. Based upon a mathematical description of the social behavior of swarms, it has been shown that this algorithm can be efficiently generated to find good solutions to a certain number of complicated situations such as, for instance, the static optimization problems, the topological optimization and others [3]-[4]-[6]-[7]. Since then, several variants of the PSO have been developed [8]-[9]-[10]-[11]-[12]-[13]-[14]. It has been shown that the question of convergence of the PSO algorithm is implicitly guaranteed if the parameters are adequately selected [15]-[16]-[4]. Several kinds of problems solving start with computer simulations in order to find and analyse the solutions which do not exist analytically or specifically have been proven to be theoretically intractable.

The particle swarm treatment supposes a population of individuals designed as real valued vectors - particles, and some iterative sequences of their domain of adaptation must be established. It is assumed that these individuals have a social behavior, which implies that the ability of social conditions, for instance, the interaction with the neighbourhood, is an important process in successfully finding good solutions to a given problem.

The strategy of the PSO algorithm is summarized as follows: We assume that each agent (particle) i can be represented in a N dimension space by its current position $X_i = (x_{i1}, x_{i2}, \cdots, x_{iN})$ and its corresponding velocity. Also a memory of its personal (previous) best position is represented by, $P_i = (p_{i1}, p_{i2}, \cdots, p_{iN})$ called (pbest), the subscript i range from 1 to s, where s indicates the size of the swarm. Commonly, each particle localizes its best value so far (pbest) and its position and consequently identifies its best value in the group (swarm), called also (sbest) among the set of values (pbest).

The velocity and position are updated as.

$$v_{ij}^{k+1} = w_{ij} v_{ij}^{k} + c_1 r_1^k [(pbest)_{ij}^{k} - x_{ij}^{k}] + c_2 r_2^k [(sbest)_{ij}^{k} - x_{ij}^{k}] \ . \tag{1}$$

$$x_{ij}^{k+1} = v_{ij}^{k+1} + x_{ij}^{k} \ . \tag{2}$$

where x_i^{k+1}, v_i^{k+1} are the position and the velocity vector of particle i respectively at iteration $k+1$, c_1 and c_2 are acceleration coefficients for each term exclusively situated in the range of 2–4, w_j is the inertia weight with its value that ranges from 0.9 to 1.2, where as r_1, r_2 are uniform random numbers between zero and one. For more details, the double subscript in the relations (1) and (2) means that the first subscript is for the particle i and the second one is for the dimension j. The role of a suitable choice of the inertia weight w_j is important in the success of the PSO. In the general case, it can be initially set equal to its maximum value,

and progressively we decrease it if the better solution is not reached. Too often, in the relation (1), v_{ij} is replaced by v_{ij}/σ where σ denotes the constriction factor that controls the velocity of the particles. This algorithm is successively accomplished with the following steps [17]-[18]-[19]:

1. Set the values of the dimension space N and the size s of the swarm (s can be taken randomly).
2. Initialize the iteration number k (in the general case is set equal to zero).
3. Place every agent between a and b; arrange them in ascending sequence respectively. There are $N+2$ nodal points and $N+1$ segments, then calculate the fitness values at the $N+1$ segments. The fitness is defined as:

$$f(i) = \sum_{j=1}^{N+1} (L_{ij} - l_{ij})(x_{ij} - x_{ij-1}) \ . \tag{3}$$

   If the termination condition is met, then stop (The termination condition is defined as: choose an ε which is very close to 0, if the minimum fitness value is less than εthen stop), choose the optimum solution $X_*(a = x_{*0} < x_{*1} < \cdots < x_{*N} < x_{*N+1} = b)$ and then

$$\int_a^b f(x)dx \simeq \sum_{j=1}^{N+1} f(x_j)(x_{*j} - x_{*j-1}) \ . \tag{4}$$

   Otherwise, continue.
4. Each agent must be updated by applying its velocity vector and its previous position using (2).
5. Repeat the above step (3, 4 and 5) until convergence criterion is reached.

The PSO algorithm is applied, with parameter setting (Table 1).

Table 1. Parameters Setting to generate the PSO Algorithm for this study

	Example1	Example2
Population Size	21	21
Number of Iterations	500	600
Acceleration Coefficients: c_1 and c_2	0.5	0.5
Inertial Weight	1.2 to 0.4	1.2 to 0.4
Desired Accuracy	10^{-5}	10^{-7}

To validate the feasibility and validity of the algorithm for numerical integration, here are some simulation examples.

3 Examples

These examples can be viewed as typical cases which provide a good illustration. We note that, the accuracy of results depends manifestly to success of particles in the swarm to locate the best points. For easy interpretation, the numerical results evaluated by PSO algorithm, and those obtained by the trapezoidal rule have been compared. The best partition generated by PSO are displayed in Tables (Table 2) in the cases $N = 5$ for $\int_{-5}^{5} \frac{dx}{1+x^2}$ and (Table 3) $N = 7$ for $\int_0^4 400x(1-x)e^{-x}dx$.

3.1 Example 1

The exact numerical value of the integral $\int_{-5}^{5} \frac{dx}{1+x^2}$ is 2.7468. Here using PSO, let N=5, $X_i = (-5 = x_{i0}, \overbrace{x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}}^{pso}, x_{i6} = 5)$, we find the error $1.6420 \cdot 10^{-7}$, as seen in Table (Table 2).

Table 2. A partition $X*$ describing the best partition X generated by PSO algorithm for the example 1

i	X_*	$f(X_*)$	$\lvert x_{*j} - x_{*j-1}\rvert$
0	-5		
1	-3.1000	0.094251	1.9000
2	-1.5000	0.307690	1.6000
3	0.0000	1.000000	1.5000
4	2.0000	0.200000	2.0000
5	3.2500	0.086486	1.2500
6	5.0000	0.038462	1.7500

$\int_{-5}^{5} \frac{dx}{1+x^2} \simeq \sum_{j=1}^{6} f(x_j)(x_{*j} - x_{*j-1}) = 0.094251 \cdot 1.9000 + 0.307690 \cdot 1.6000 + 1.0000 \cdot 1.5000 + 0.20000 \cdot 2.0000 + 0.086486 \cdot 1.2500 + 0.038462 \cdot 1.7500 = 2.7468$
The error using the trapezoidal rule for $N = 5$ is 0.26988

3.2 Example 2

The exact numerical value of the integral $\int_0^4 400x(1-x)e^{-x}dx$ is 1.0735. Here using PSO, let, $N = 7$ $X_i = (0 = x_{i0}, \overbrace{x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{i6}, x_{i7}}^{pso}, x_{i8} = 4)$, we find the error $4.1683 \cdot 10^{-7}$, as seen in Table (Table 3),

$\int_0^4 400x(1-x)e^{-x}dx \simeq \sum_{j=1}^{8} f(x_j)(x_{*j} - x_{*j-1}) = 45.2940 \cdot 0.34652 + 30.0970 \cdot 0.23900 - 9.8074 \cdot 0.64908 - 14.653 \cdot 0.76540 - 2.1595 \cdot 1.79000 - 2.0152 \cdot 0.05000 - 1.6922 \cdot 0.12486 - 1.6102 \cdot 0.035140 = 1.0731$. The error using the trapezoidal rule for N=80 is $8.2775 \cdot 10^{-2}$(incomparable!).

Table 3. A partition $X*$ describing the best partition X generated by PSO algorithm for the example 2

i	X_*	$f(X_*)$	$\lvert x_{*j} - x_{*j-1}\rvert$
0	0.00000		
1	0.34650	45.2940	0.34652
2	0.58552	30.0970	0.23900
3	1.23460	-9.8074	0.64908
4	2.00000	-14.653	0.76540
5	3.79000	-2.1595	1.79000
6	3.84000	-2.0152	0.05000
7	3.96490	-1.6922	0.12486
8	4.00000	-1.6102	0.03514

4 Conclusion

The particle swarm optimization is used to investigate the best integrating points. Some good results are obtained by using the specific PSO approach. It is now known that the PSO scheme is powerful, and easier to apply specially for this type of problems. Also, the PSO method can be used directly and in a straightforward manner.

Acknowledgement

This work was sponsored, in part, by the MERS (Ministère de l'Enseignement et de la recherche Scientifique): Under the contract No B*0142/0/0016.

References

1. Wang, X.-h., et al.: Numerical Integration Study Based on Triangle Basis Neural Network Algorithm. Journal of Electronics and Information Technology 26(3), 394–399 (2004)
2. Eberhart, R.C., Kennedy, J.: A new optimizer using particles swarm theory. In: Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
3. Parsopoulos, K.E., Vrahatis, M.N.: Modification of the Particle Swarm Optimizer for Locating all the Global Minima. In: Kurkova, V., et al. (eds.) Artificial Neural Networks and Genetic Algorithms, pp. 324–327. Springer, New York (2001)
4. Parsopoulos, K.E., et al.: Stretching technique for obtaining global minimizers through particle swarm optimization. In: Proc. of the PSO Workshop, Indianapolis, USA, pp. 22–29 (2001)
5. Parsopoulos, K.E., et al.: Objective function stretching to alleviate convergence to local minima. Nonlinear Analysis TMA 47, 3419–3424 (2001)
6. Fourie, P.C., Groenwold, A.A.: Particle swarms in size and shape optimization. In: Proceedings of the International Workshop on Multi-disciplinary Design Optimization, Pretoria, South Africa, August 7-10, pp. 97–106 (2000)

7. Fourie, P.C., Groenwold, A.A.: Particle swarms in topology optimization. In: Extended Abstracts of the Fourth World Congress of Structural and Multidisciplinary Optimization, Dalian, China, June 4-8, pp. 52–53 (2001)
8. Eberhart, R.C., et al.: Computational Intelligence PC Tools. Academic Press Professional, Boston (1996)
9. Kennedy, J.: The behaviour of particles. Evol. Progr. VII, 581–587 (1998)
10. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
11. Shi, Y.H., Eberhart, R.C.: Fuzzy adaptive particle swarm optimization. In: IEEE Int. Conf. on Evolutionary Computation, pp. 101–106 (2001)
12. Shi, Y.H., Eberhart, R.C.: A modified particle swarm optimizer. In: Proc. of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9 (1998)
13. Shi, Y.H., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)
14. Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, Washington DC, pp. 1951–1957 (1999)
15. Eberhart, R.C., Shi, Y.: Parameter selection in particle swarm optimization. In: Porto, V.W. (ed.) (1998)
16. Cristian, T.I.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317–325 (2003)
17. Zerarka, A., Khelil, N.: A generalized integral quadratic method: improvement of the solution for one dimensional Volterra integral equation using particle swarm optimization. Int. J. Simulation and Process Modelling 2(1-2), 152–163 (2006)
18. Zerarka, A., Soukeur, A., Khelil, N.: The particle swarm optimization against the Runge's phenomenon: Application to the generalized integral quadrature method. International Journal of Mathematical and Statistical Sciences 1(3), 171–176 (2009)
19. Khelil, N., et al.: Improvement of Gregory's formula using Particle Swarm Optimization. In: The proceeding of International Conference on Computer and Applied Mathematics, vol. 58, pp. 940–942 (2009)

Identification of VSD System Parameters with Particle Swarm Optimization Method

Yiming Qiu, Wenqi Li, Dongsheng Yang, Lei Wang, and Qidi Wu

School of Electronics and Information Engineering, TongjiUniversity,
Shanghai 200092, China
hans.qiu@keb.cn, robert.li@analog.com,
leojyang@hotmail.com, wanglei_tj@126.com

Abstract. A VSD system, which consists of an inverter & an induction motor, is now widely used in all kinds of application. But from the view point of an end user, neither the motor parameters in the mathematics model nor the vector controller structure are known. In this paper a PSO algorithm is programmed with IEC61131-3 language to estimate the parameters for the VSD system, based on the hardware of a vector controlled inverter, in order to reach the similar dynamic performance as a DC motor system. The PSO algorithm could be a kind of alternative approach of present parameter identification functions, for its requirements on the speed of CPU and volume of memory are low, while it converges quickly. It's especially helpful for the adjustment of complicated control system, when the technical requirements are clear & measurable.

Keywords: PSO, VSD, Induction Motor, Parameter Identification.

1 Introduction

In recent years, instead of DC motor, induction motor became more popular in a high precise and high performance transmission system, because of its simple, reliable, and robust structure. In order to get the similar performance as a DC system, for field oriented vector control, a precise motor model & vector control structure model are the prerequisite for high performance. But in an actual AC VSD system, these two models are just like black box to the user, which limits the actual performance of the system.

There are lots of suppliers, which produce vector-controlled inverter with the latest CPU & DSP technology. About the control structure, it is really difficult for someone else, except the designer himself, to use existing mathematical methods to make an accurate description, because it is not only vector control method, but also a combination of feed-forward, feed-back, interpolation, compensation and other engineering methods. Being a user, it is impossible to know the actual control algorithm. So in most cases, the internal mathematical model and the control algorithm of the inverter are complete "Black Box", which input can be changed, which output can be observed, but the internal is invisible.

There are differences between the parameters in the nameplate and the real motor parameters too. Because most induction motors are originally designed for open-loop applications. So there is no concern on the characteristics, and in the nameplate only

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 227–233, 2011.
© Springer-Verlag Berlin Heidelberg 2011

rated frequency and synchronous speed are marked, instead of the rated speed (or rated slip). In the same time, there are lots of specially designed motors without detailed motor parameters to prevent a copy from the competitors. Because of the material differences in the manufacturing process, there could be a deviation between the real motor parameter and the designed value;

Before this study work, several kinds of motor parameter identification methods have been discussed already. Most scholars finished their identification of the motor mathematical model in a pure simulation environment, compared recognition results with the "precision motor parameters", and compared a variety of different identification methods on identification value, error, identification rate, to prove the validity of the algorithm [1]. Or first put different forms of voltage and current signals on a motor, and detected the motor voltage & current, then established the motor mathematical model, put the same forms of voltage and current signals to the model, and measured the output response, which should be similar to the measured value for the tested motor if the motor parameters were right. The recognition results were similar to the induction motor parameters obtained from the standard way, which is to perform a locked rotor and a no-load test [2]. All these studies only focused on the motor itself without considering the function of frequency inverter. For end users, the combination of motor and frequency inverter is significantly to be treated as a black box together.

Based on the existing hardware and software, a particle swarm optimization algorithm is introduced in this paper to make iterative searching for the correct parameters for the VSD system, by comparing the performance to a DC motor system. This method is doubtless to be a rare substitute of the original inverter functions.

2 Particle Swarm Optimization Algorithm [3][4]

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking. In PSO, all of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. After each time of iteration, every particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value, which is tracked by the particle swarm optimizer, is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. After finding the two best values, the particle updates its velocity and position with following equation (1) and (2):

$$V_{ik}(t+1) = \omega \cdot V_{ik}(t) + c_1 \cdot rand_1(p_{ik\,best}(t) - x_{ik}(t)) + c_2 \cdot rand_2(g_{kbest}(t) - x_{ik}(t)) \quad . \tag{1}$$

$$x_{ik}(t+1) = x_{ik}(t) + V_{ik}(t+1) \quad . \tag{2}$$

V_{ik}(t+1)is the particle's velocity, x_{ik}(t+1)is the current particle's position. p_{ikbest}(t) and gkbest(t) are defined as stated before. ω is weighting factor, c1, c2 are learning

factors, $rand_1$&$rand_2$ are two random number between (0,1). $V_{ik}(t+1) \in [-V_{max}, V_{max}]$, Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max}, then the velocity on that dimension is limited to V_{max}. $x_{ik}(t+1) \in [-x_{max}, x_{max}]$, Particles' positions should be in a certain range, which might be fit to their physic value.

3 Inverter Parameters and Motor Nameplate Parameters [5] [6]

An inverter 09.F5.A1D-2BDA from company KEB is used in this research. In the application manual, a standard vector control algorithm is described. Its performance has been proved in lots of application in last decade. But the internal control algorithm is really magical, for only stator resistor R_s and leakage inductance L_Σ are needed for the motor equivalent model. In closed loop control mode, even these two parameters are not so sensible. In order to reach an excellent performance, the right motor nameplate data is needed.

Table 1. Nameplate parameters of an induction motor in this research

U_{rated} (V)	I_{rated} (A)	f_{rated} (Hz)	n_{rated} (rpm)	cos(phi)	P_{rated} (kw)
220	0.87	50	3000	0.84	0.18

Normally this kind of nameplate is not enough for a VSD system. We can only treat the following three parameters P_{rated}, I_{rated}, f_{rated} as precise value. Obviously the listed value 3000rpm is only a synchronous speed at rated frequency. The list value 220V is only the voltage of net. *Cos*(phi) will influence the magnetizing current. And the inverter parameter, Field weakening factor F_2, will influence the performance of VSD system in field weakening zone. All these 4 parameters are very sensitive for the performance of a VSD system. There are unknown relationships between them. An auto-identification of these 4 parameters will be valuable in engineering level.

Table 2. VSD system parameters to be identified

No.	Parameter	Physical range	Identified range
1	n_{rated}	1… 64000 rpm	2500…2999
2	U_{rated}	120…830v	120…300
3	cos(phi)	0.50…1.00	50…100
4	F_2	0.01…3.00	1…300

4 Objective Functions

We know that any power electric drive systems are subject to the basic equation of motion:

$$M_e - M_L = \frac{J}{n_p}\frac{d\omega}{dt} \quad . \tag{3}$$

Only with a complete decoupling control, one induction motor can achieve the comparable performance as DC motor. Therefore the speed characteristics of a DC motor can be used to measure the fitness value of present motor parameters. We keep the same ramp time for accelerating and decelerating, if the following conditions can be fitted to a certain level, we can say that the system model is almost fully decoupled.

Condition 1: the speed deviation at accelerating & decelerating should be as small as possible.

$$F_n = \sqrt{\frac{\sum_{i=1}^{n} \left(n_{set}(i) - n_{act}(i)\right)^2}{n}} \quad . \tag{4}$$

Condition 2: the actual motor torque should be constant with the constant accelerating rate up to field weakening range. And the ripple of actual torque should be as small as possible.

$$F_M = \sqrt{\frac{\sum_{i=1}^{n} \left(M_{act}(i) - \overline{M}_{act}\right)^2}{n}} \quad . \tag{5}$$

Condition 3: The sum of accelerating torque and decelerating torque should be two times of the friction torque. This can be set manually to improve the precision of the identification.

$$F_{Mf} = \left| \frac{\overline{M}_{acc} + \overline{M}_{dec}}{2} - M_f \right| \quad . \tag{6}$$

Combining these 3 conditions, we get following objective function.

$$F = F(F_n, F_M, F_{Mf}) \quad . \tag{7}$$

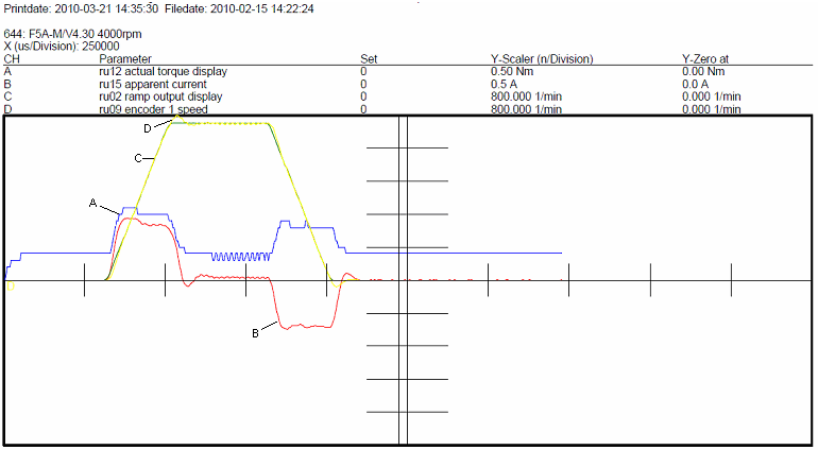


Fig. 1. Optimized running curve.(A: torque, B: current, C: set-speed, D: actual speed)

In Fig.1 we can find a running curve of an AC VSD system, which is comparable with a DC motor system. In this case, the objective function value will be very low.

5 Test System&Algorithm

The test system consists of an induction motor, a vector-control inverter, a motion controller, and a desktop PC. The PSO algorithm is programmed in a motion controller with IEC61131-3. After each flying, the new motor parameters are sent from the motion controller to inverter, and an auto-tuning of the control structure is activated according to these motor data. Then the motion controller sends out commands to the inverter to control the motor to follow a designed speed profile, accelerating, running constant and

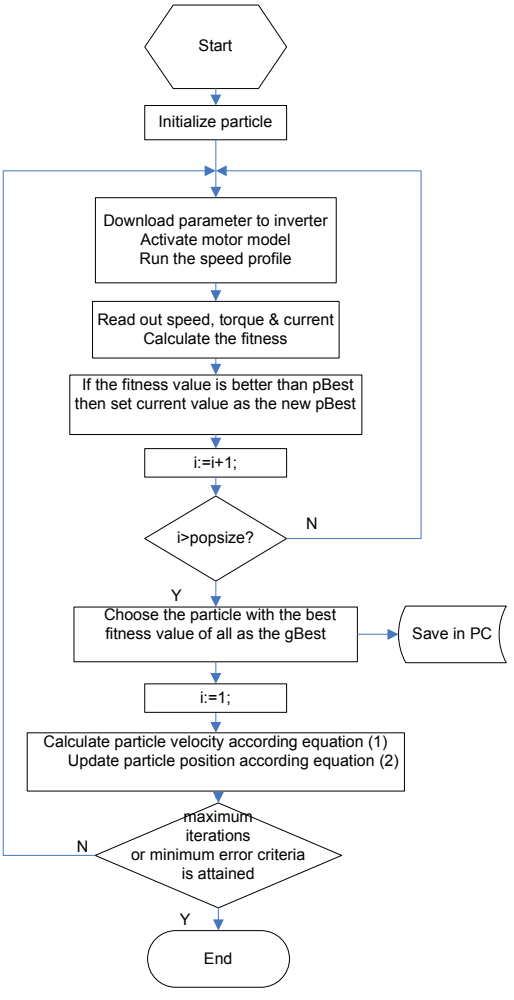


Fig. 2. Procedure of PSO

decelerating to zero speed. The actual values are sent back to motion controller to calculate the fitness value. The PC reads out and records the data of each particle after each flying through MODBUS TCP with100BASE-TX. The position, velocity, fitness value, pbest of each particle and gbest are included.Procedure of PSO based control algorithm is listed in Fig.2.

6 Identification and Results

We tried PSO algorithm to identify the following 4 parameters of this motor: $n_{rated,}$ $U_{rated,}$ cos(phi) and field weakening factor F_2. 17 particles were used. After 11 times of iteration, the objective value was acceptable with gbest [2543,206,85,20].

Table 3. Nameplate parameter & identified value

Parameter	n_{rated}(rpm)	U_{rated}(V)	cos(phi)	F2
Nameplate	3000	220	0.84	1.2(Default)
Identified	2543	206	0.85	0.2

When we put the identified parameter value into the inverter, and run the VSD system to follow the defined speed profile, the dynamical performance can meet the defined conditions of objective functions. It's closed to the performance of a DC system. We optimized the objective functions and PSO algorithm during the identification, and the motor became hot. So these values are fit for a hot motor. This is what we need for a real application. The definition of objective functions and the PSO algorithm were proved.

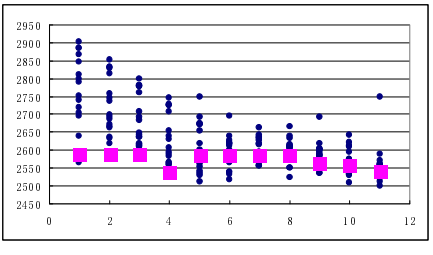


Fig. 3. Parameter_n_{rated}

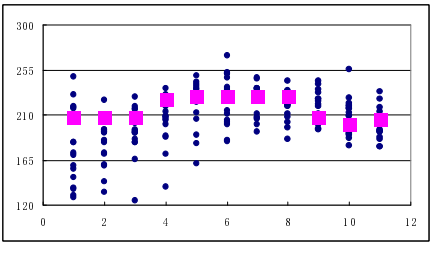


Fig. 4. Parameter_U_{rated}

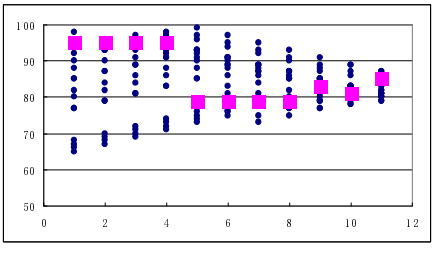


Fig. 5. Parameter_cos(phi)

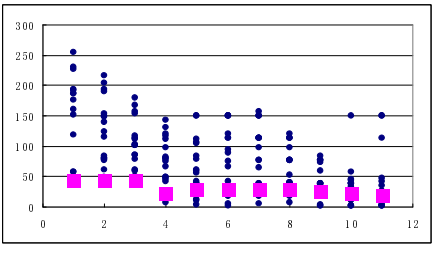


Fig. 6. Parameter_F_2

Fig.3 to Fig.6 shows the 4 dimensions of the positions of 17 particles&gbest particle after each time of iteration.During the optimization, we found that the performances with several pbest solutions were quite closed to final solution. They could be accepted by most application too.

7 Summary

In this paper, PSO algorithm is used to identify the VSD system parameters. The motor and inverter both are treated as a black box. By comparing the dynamic performance with DC motor characteristics in following a given speed profile, the fitness of motor nameplate parameters and system parameter F_2 are calculated. PSO algorithm is verified. It could be a new optimization method for parameter-identificationin an unknown system.

Acknowledgments

This research is sponsored by National Natural Science Foundation of China (NSFC)（70871091, 61075064, 61034004）, Ph.D. Programs Foundation of Ministry of Education of China (20100072110038), Foundation of the Ministry of Education of China for Returned Scholars, Program for New Century Excellent Talents in University of Ministry of Education of China.

References

1. Picardi, C., Rogano, N.: Parameter identification of induction motor based on particle swarm optimization. In: International Symposium on Power Electronics, Electrical Drives, Automation and Motion, SPEEDAM 2006, pp. 968–973 (2006)
2. Karimi, A., Choudhry, M.A., Feliachi, A.: PSO-based Evolutionary Optimization for ParameterIdentification of an Induction Motor. In: 2007 39th North American Power Symposium (NAPS 2007), pp. 659–664 (2007)
3. Kennedy, J., Eberhart,R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, pp. 1942–1948 (1995)
4. Wu, Q.D., Wang, L.: Intelligent Particle Swarm Optimization Algorithm Research and Application. Jiangsu Education Publishing House, Nanjing (2005) (in Chinese)
5. Boshi, C.: Electric Drive & Automatic Control System. Department of Automation. Shanghai University of Industrial (1991) (in Chinese)
6. Brinkmann, K.E.: GmbH: Application Manual of COMBIVERT F5-MULTI / SERVO 2.8, Art.Nr.: 00.F5.MEA-K280 (2005)

PSO-Based Emergency Evacuation Simulation

Jialiang Kou, Shengwu Xiong*, Hongbing Liu, Xinlu Zong, Shuzhen Wan, Yi Liu, Hui Li, and Pengfei Duan

School of Computer Science and Technology,
Wuhan University of Technology,
122 Luoshi Road, Wuhan, Hubei, China, 430070
koujialiang@yahoo.com.cn, swxiong@gmail.com, 329323460@qq.com, zongxinlu@sina.com, wanshuzhen@163.com, 25101654@qq.com, lipeilin1984xyz@163.com, 435887385@qq.com

Abstract. The Emergency Evacuation Simulation (EES) has been increasingly becoming a hotspot in the field of transportation. PSO-based EES is a good choice as its low computation complexity compared with some other algorithms, especially in an emergency. The selection of fitness function of each particle in PSO is a key problem for EES. This paper will introduce some fitness functions for EES and present a new fitness function called Triple-Distance Safe Degree (TDSD). Through theoretical analysis and experimental validation, the TDSD is proved to be much better than other fitness functions introduced in this paper.

Keywords: PSO; Emergency Evacuation Simulation; Triple-Distance Safe Degree.

1 Introduction

Emergency evacuation simulation (EES) has been increasingly becoming a hotspot [1] [2] [3] in the field of transportation, especially recent ten years, with occurrences of more and more emergency incidents, such as September 11 Attacks [4], Wenchuan Earthquake [5], Hurricane Katrina [6] and so on. PSO is originally proposed by Kennedy, Eberhart and Shi [7] [8] [9]. The PSO is a kind of optimization method by iteratively improving candidate solutions according to given fitness function. On the other hand, the PSO could be used as a simulation method to simulate a social or natural procedure in real life. There have been some researches on PSO based EES at recent years. Yang et al. [10] [11] proposed a new emergency evacuation model based on Multi-Agent framework and Linear Weight Decreasing PSO to simulate human behavior during emergency evacuation in continuous indoor small space. Yusoff et al. [12] utilized a discrete PSO to simulate evacuation. Izquierdo et al. [13] established a model for evacuation time estimation and behavioral patterns assessment. Actually, it

* Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 234–241, 2011.
© Springer-Verlag Berlin Heidelberg 2011

is a simulation and assessment system. Although the previous works have investigated some problems on PSO based EES, neither of them studied the fitness function systematically. On the basis of the former studies of the predecessors, this paper will briefly introduce a kind of PSO-based EES method and concentrate on the fitness selection problem of PSO based EES. After analyzing two sorts of fitness function, this paper presents a novel fitness function called TDSD. The structure of this paper is as follows.

This paper starts with introducing the situation of PSO-based EES researches and the main focus of this paper. And then, it introduces the PSO-based EES method employed in this paper. In the third part, it focuses on the fitness selection problem and proposes a completely new fitness function called TDSD. At last, the TDSD is tested through experiments and the conclusion is drawn out.

2 The Abstractions and Representations

2.1 The Scenario

A stadium scenario is selected as the scenario for this study. The stadium is abstracted as an undirected graph. The edges of the graph represent the roads inside and outside the stadium. The vertexes of the graph represent the intersections of the roads. Fig.1 shows the stadium scenario.

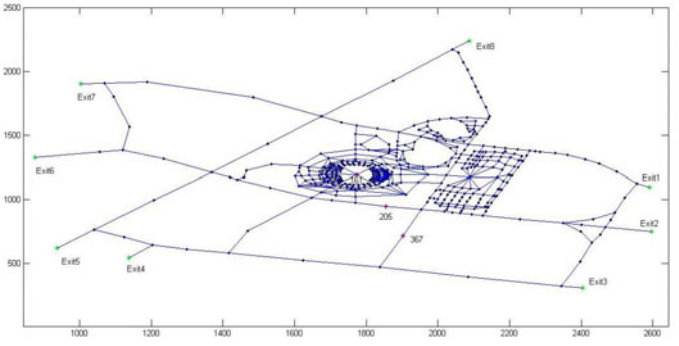


Fig. 1. The scenario of a stadium. The blue lines represent the roads inside and outside the stadium. The black dots represent the intersections of the roads. The green dots represent the exits. There are 8 exits in this scenario. The red dots represent the danger points.

2.2 The Representation of Evacuation Solution

The evacuation solution is represented as a matrix shown as equation 1. The jth column of the matrix represents the jth person's route. The ith row represents the ith time step. The v_{ij} represents the position of the jth person on the ith time step. The R_j represents the jth person's route. The m is the number of time steps of the last person evacuated from the stadium. The n is the total number of people.

$$S = \begin{bmatrix} v_{11} & v_{12} & v_{13} & \cdots & \cdots & v_{1n} \\ v_{21} & v_{22} & v_{23} & \cdots & \cdots & v_{2n} \\ v_{31} & v_{32} & v_{33} & \cdots & \cdots & v_{3n} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ v_{m1} & v_{m2} & v_{m3} & \cdots & \cdots & v_{mn} \end{bmatrix} = \begin{bmatrix} R_1 & R_2 & \cdots & R_n \end{bmatrix} \quad (1)$$

2.3 The PSO Based Emergency Evacuation Simulation

Each person is seen as a particle in PSO. The fitness of a particle represents the safe degree of a person. The main procedure of the PSO for evacuation simulation is as follows:

1. Generate the danger points randomly.
2. Set the exits of the stadium.
3. Set the study factor and inertia weight of PSO.
4. Initialize each particle's position randomly.
5. Initialize the real speed of each person randomly. The real speed refers to the real running speed in evacuation here.
6. Initialize the velocity of each particle randomly.
7. Do the following loop, till all the particles arrive at the exits.

```
Function PSO_EES()
  For t=1:M %M is the maximum number of time steps
      For i=1:N %N is the number of particles
          If RemainingDistance(i)>MinRemainingDistance
             Keep current velocity unchanged;
             Keep current position unchanged;
             RemainingDistance(i)=RemainingDistance(i)—
             RealSpeed(i);
             RealSpeed(i)=CalculateRealSpeed();
          Else
              Change velocity according to PSO iteration
              formulas;
              Change Position according to PSO iteration
               formulas;
              RemainingDistance(i)=RemainingDistance(i)-
              RealSpeed(i);
              RemainingDistance(i)=RemainingDistance(i)+
              Distance(current_position,next_position);
              RealSpeed(i)= CalculateRealSpeed();
          End
```

```
            If Fitness(Position(i))>
               Fitness(Local_Best_Position(i))
                Local_Best_Position(i)=Position(i);
            End
        End
        For i=1:N
             If Fitness(Local_Best_Position(i))>
                Fitness(Global_Best_Position)
                 Global_Best_Position= Local_Best_Position(i);
             End
         End
        For i=1:N
            Position_Record(t,i)=Position(i);
        End

         For i=1:NV % NV is the number of vertexes
             NPV(i)=0;
         End
        QuitFlag=1;
        For i=1:N
            If Position(i) is not one of Exits
               NPV(Position(i))=NPV(Position(i))+1;
               ExitFlag=0;
            End
        End
        If QuitFlag==0
           Quit();
        End
    End
End
```

8. Record the generated evacuation solution.

3 The Fitness Functions of PSO for EES

The selection of fitness function for EES is the key problem which affects the validity of the results of the simulation. In other words, the results of simulation how close to the reality is largely depended on the fitness function.

3.1 The Distance to the Danger Point (DDP)

The Distance to the Danger Point refers to the distance between the person and the nearest danger point to the person. The definition of the DDP [14] is as equation 2.

$$DDP = \min\left(\sqrt{\left(X_p - X_{D_i}\right) + \left(Y_P - Y_{D_i}\right)}\right)_{i=1}^{NDP} \tag{2}$$

$$Fitness = \frac{\left(DDP\right)^{0.8}}{300} \tag{3}$$

The NDP means the Number of Danger Points. X_p and Y_p are the horizontal and vertical coordinate of position of the person. X_{Di} and Y_{Di} are the horizontal and vertical coordinate of the ith danger point. Obviously, the lower the DDP, the lower the fitness is. But, the lower the DDP, the more dangerous the person is?

As shown in Fig.2, hypothesize that there is only one exit Exit4 and one danger point Intersection369 in the stadium. Although the Intersection272 is farther than the Intersection371 away from the danger point, the Intersection371 is safer than the Intersection272. That is because that the Intersection371 is nearer than the Intersection272 to the Exit4. Obviously, it is not suitable to measure the safe degree just employing the DDP here.

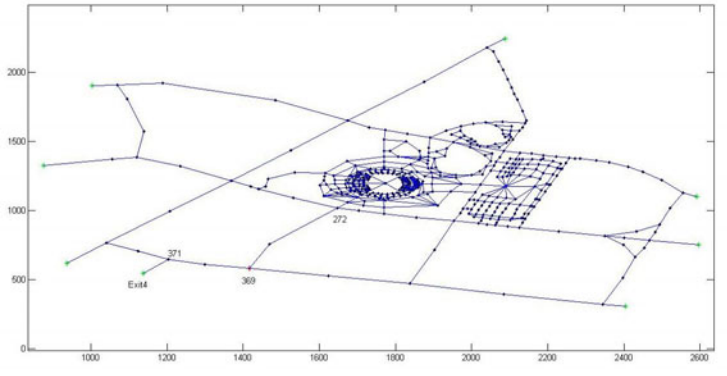


Fig. 2. An example to validate the reliability of DDP

3.2 The Distance to the Exit (DE)

The Distance to the Exit means the distance between the person and the nearest exit to the person. The DE [13] is defined as equation 4.

$$DE = \min\left(\sqrt{\left(X_p - X_{E_i}\right) + \left(Y_P - Y_{E_i}\right)}\right)_{i=1}^{NE} \tag{4}$$

$$Fitness = 1 - \frac{DE}{C}, C \text{ is a constant larger than } DE \tag{5}$$

The NE means the Number of Exits. X_p and Y_p are the horizontal and vertical coordinate of position of the person. X_{Ei} and Y_{Ei} are the horizontal and vertical coordinate of the ith danger point. Obviously, the smaller the DE, the higher the fitness is. But, the smaller the DE, the safer the person is?

As shown in Figure 3, there are two exits (Exit4 and Exit8) and only one danger point (the Intersection370). Assume that the distance between Exit4 and Intersection371 is the same as the distance between Exit8 and Intersection403. But, the Intersection371 is closer to the danger point than Intersection403. Thus, the Intersection371 is more dangerous than Intersection403. Clearly, it is not sensible merely to use DE to evaluate the safe degree in this example.

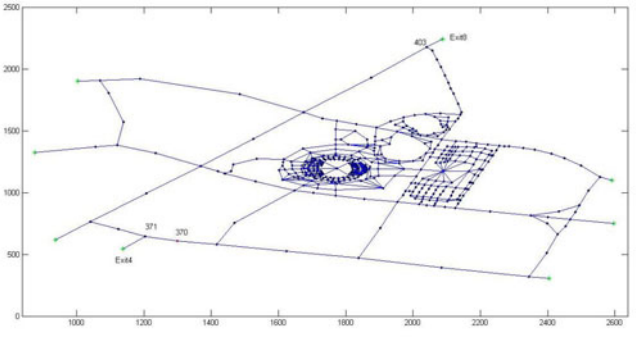


Fig. 3. An example to validate the reliability of DE

3.3 The Triple-Distance Safe Degree (TDSD)

Considering the distance relations between the exits, the danger points, the center point of the building and the person, the TDSD is defined as equation 6.

$$\begin{cases} Fitness_i = TDSD_i = \left(DC_i / C\right) * \sqrt[64]{\dfrac{4}{\pi^2} * \operatorname{arccot}\left(DE_i / C\right) * \arctan\left(DDP_i / C\right)}, \\ i = 1, 2, \cdots, NP \\ DE_i = \min\left(\sqrt{\left(X_{P_i} - X_{E_j}\right)^2 + \left(Y_{P_i} - Y_{E_j}\right)^2}\right)_{j=1}^{NE} \\ DDP_i = \min\left(\sqrt{\left(X_{P_i} - X_{D_k}\right)^2 + \left(Y_{P_i} - Y_{D_k}\right)^2}\right)_{k=1}^{NDP} \\ DC_i = \sqrt{\left(X_{P_i} - X_C\right)^2 + \left(Y_{P_i} - Y_C\right)^2} \end{cases} \tag{6}$$

C is a constant larger than DC_i, DE_i and DDP_i. DC_i is the distance between the ith person and the center point of the building. DE_i is the minimum distance between the ith person and the exits. DDP_i is the minimum distance between the ith person and the danger points. X_{pi} is the horizontal coordinate of the ith person. Y_{pi} is the vertical coordinate of the ith person. X_{Ej} is the horizontal coordinate of the jth exit. Y_{Ej} is the

vertical coordinate of the jth exit. X_{Dk} is the horizontal coordinate of the kth Danger Point. Y_{Dk} is the vertical coordinate of the kth Danger Point. X_c is the horizontal coordinate of the center point. Y_c is the vertical coordinate of the center point. NP is the number of the people. NE is the number of the exits. NDP is the number of the danger points. $Fitness_i$ is the fitness of the ith person.

The TDSD is composed of three parts. The first part is about the distance to the center point (DC) of the building. The second part is an arc cotangent function which means the safe degree relative to the DE. With the increase of DE, the safe degree goes down. The third part is an arc tangent function which means the safe degree relative to the DDP. With the increase of DDP, the safe degree goes up. The fitness of the ith person is the product of the three parts. The name TDSD originates from the combination of the DC, the DE and DDP.

4 Experiment and Analysis

As shown in Fig.1, there are 8 exits (Exit1 to Exit8) and 3 danger points (Intersection161, Intersection205 and Intersection367). As shown in Fig.4 (the left), through comparing the safe degree of 476 intersections generated by DE, DDP and TDSD with the real safe degree, it is proved that the TDSD curve could best match the real curve. As shown in Fig.4 (the right), the safe degree calculated by TDSD possess the smallest average deviation with the real value just 0.0080, much less than DDP (0.0517) and DE (0.1606). Obviously, TDSD is the best fitness function among the three types. The TDSD reveals the fact that the safe degree is not just relative with the center point, the danger points or the exits, but both of them.

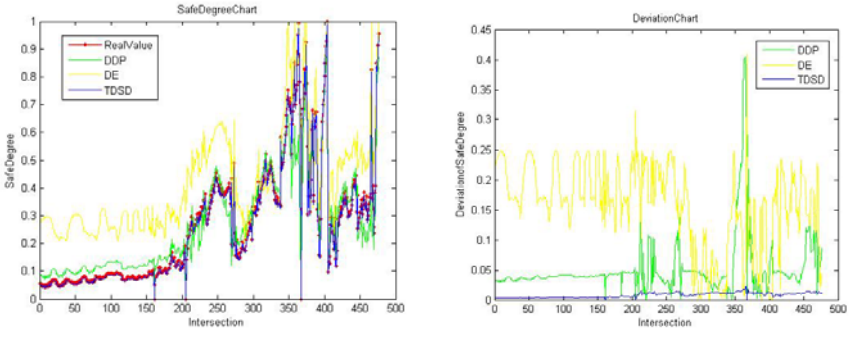


Fig. 4. The left chart is the safe degree and the right chart is the deviation between the safe degree and the real value

5 Conclusion and Future Works

This paper introduces a kind of PSO-based EES method and presents a completely new fitness function for EES. Through the comparison with other two kinds of fitness

functions in the scenario of a stadium, the TDSD is proved to meet the physical reality best in three type fitness functions. As the limitation of space, the correctness and feasibility of the TDSD in more scenarios is not discussed and tested. That will become the future direction of this study.

Acknowledgements. This work was supported in part by the National Science Foundation of China under grant no. 40971233.

References

1. Church, R.L., Cova, T.J.: Mapping evacuation risk on transportation networks using a spatial optimization model. Transportation Research Part C: Emerging Technologies 8(1-6), 321–326 (2000)
2. Lämmel, G., Grether, D., Nagel, K.: The representation and implementation of time-dependent inundation in large-scale microscopic evacuation simulations. Transportation Research Part C: Emerging Technologies 18(1), 84–98 (2010)
3. Xie, C., Turnquist, M.A.: Lane-based evacuation network optimization: An integrated Lagrangian relaxation and tabu search approach. Transportation Research Part C: Emerging Technologies 19(1), 40–63 (2011)
4. September 11 attacks, http://en.wikipedia.org/wiki/September_11_attacks
5. 2008 Sichuan earthquake, http://en.wikipedia.org/wiki/Wenchuan_earthquake
6. Katrina, H.: http://en.wikipedia.org/wiki/Katrina_Hurricane
7. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: The 1995 IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE, Piscataway (1995)
8. Kennedy, J.: The particle swarm: social adaptation of knowledge. In: The 1997 IEEE International Conference on Evolutionary Computation, pp. 303–308. IEEE, Piscataway (1997)
9. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: The 1998 IEEE International Conference on Evolutionary Computation, pp. 69–73. IEEE, Piscataway (1998)
10. Yang, B., Wang, C., Huang, H., Li, L.: A Multi-Agent and PSO Based Simulation for Human Behavior in Emergency Evacuation. In: The 2007 International Conference on Computational Intelligence and Security, pp. 295–300. IEEE Computer Society, Piscataway (2007)
11. Yang, B., Wang, C., Li, L., Huang, H.: A Modified Particle Swarm Optimization-based Human Behavior Modeling for Emergency Evacuation Simulation System. In: The 2008 IEEE International Conference on Information and Automation, pp. 23–28. IEEE Computer Society, Piscataway (2008)
12. Yusoff, M., Ariffin, J., Mohamed, A.: An Improved Discrete Particle Swarm Optimization in Evacuation Planning. In: 2009 International Conference of Soft Computing and Pattern Recognition, pp. 49–53 (2009)
13. Izquierdo, J., Montalvo, I., Pérez, R., Fuertes, V.S.: Forecasting pedestrian evacuation times by using swarm intelligence. Physica A: Statistical Mechanics and its Applications 388(7), 1213–1220 (2009)
14. Zeng, S., Ma, X.Q., Liao, Y.F.: The Research of Evacuation Model in Urban Underground Business Buildings. Building Science 24(5), 27–32 (2008)

Training Spiking Neurons by Means of Particle Swarm Optimization

Roberto A. Vázquez[1] and Beatriz A. Garro[2]

[1] Intelligent Systems Group
Faculty of Engineering - La Salle University
Benjamín Franklin 47 Col. Condesa CP 06140 México, D.F.
[2] Center for Computing Research - IPN
Av. Juan de Dios Batiz sn, Col. Nueva Industrial Vallejo, CP 07738 México, D.F.
ravem@lasallistas.org.mx, bgarrol@ipn.mx

Abstract. Meta-heuristic algorithms inspired by nature have been used in a wide range of optimization problems. These types of algorithms have gained popularity in the field of artificial neural networks (ANN). On the other hand, spiking neural networks are a new type of ANN that simulates the behaviour of a biological neural network in a more realistic manner. Furthermore, these neural models have been applied to solve some pattern recognition problems. In this paper, it is proposed the use of the particle swarm optimization (PSO) algorithm to adjust the synaptic weights of a spiking neuron when it is applied to solve a pattern classification task. Given a set of input patterns belonging to K classes, each input pattern is transformed into an input signal. Then, the spiking neuron is stimulated during T ms and the firing rate is computed. After adjusting the synaptic weights of the neural model using the PSO algorithm, input patterns belonging to the same class will generate similar firing rates. On the contrary, input patterns belonging to other classes will generate firing rates different enough to discriminate among the classes. At last, a comparison between the PSO algorithm and a differential evolution algorithm is presented when the spiking neural model is applied to solve non-linear and real object recognition problems.

1 Introduction

James Kennedy and Russell C. Eberhart proposed the particle swarm optimization (PSO) algorithm in 1995 [11]. This algorithm is a method for combinatorial or numerical optimization problems. Several versions of this algorithm have been proposed in the last years; however, the principal characteristic of this algorithm is that it can optimize over non-linear and non-continuous search spaces. In addition, this PSO algorithm is based on the collective behavior of the bird flocking, fish schooling. This behavior consists on moving to another better place for eating or getting a better life.

The PSO algorithm has been applied in a wide range of problems, including those related to the field of artificial neural networks (ANN). These types of algorithms are non-gradient approaches; this characteristic makes them a promising

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 242–249, 2011.
© Springer-Verlag Berlin Heidelberg 2011

learning strategy for the training (adjusting the synaptic weights) of an ANN [12], [13] and [14]. Furthermore, the PSO algorithm has been applied to design automatically feed-forward neural networks; this includes the design of the best topology, the transfer function of each neuron and the synaptic weights [1].

One of the most common applications of an ANN is a pattern classification task. However, though these models have been applied to solve several pattern recognition problems, they have some constrains that limits their applicability in real-life problems, particularly during the design of the ANN. For that reason, it is mandatory to explore new ways of implementing ANN.

Recently, a new type of neural model, called spiking neuron models, emerged. These models have been called the 3rd generation of artificial neural networks [2]. The spiking neuron models increase the level of realism in a neural simulation and incorporate the concept of time. They also have been applied in a wide range of areas from the field of computational neurosciences such as: brain region modeling, auditory processing [3], visual processing [4], robotics [5] and so on. Nonetheless, their application for solving pattern recognition problems is gaining popularity in the field of artificial intelligence.

In [17], the authors described how a genetic algorithm can be used during the learning process of a spiking neural network; however, the model was not applied to perform a pattern recognition problem. In [16], the authors show how a spiking neural network can be trained by a quantum PSO and then applied to a string pattern recognition problem. In [18] the authors trained a spiking neural network using a PSO algorithm, and then it is applied in three pattern recognition problems; however, in each problem, the authors had to design the topology of the network. In [10,15,19], the authors shown how only one spiking neuron such as Leaky-Integrate-and-Fire and Izhikevich models [6,7] can be applied to solve different linear and non-linear pattern recognition problems.

In this paper is shown how it is possible to apply a spiking neuron, trained with a PSO algorithm, in different linear and non-linear pattern recognition problems. Given a set of input patterns belonging to K classes, each input pattern is transformed into an input signal. After that, the spiking neuron is stimulated during T ms and the firing rate is computed. Once the synaptic weights of the neuron model were adjusted by the PSO algorithm, we expect that the input patterns which belong to the same class generate similar firing rate. On the contrary, patterns belonging to other classes generate firing rates different enough to discriminate among the classes. Finally, the proposed method is compared against the method described in [19] when the spiking neuron is applied to solve pattern recognition problems.

2 Spiking Neuron Model

A typical spiking neuron can be divided into three functionally distinct parts, called dendrites, soma, and axon. The dendrites play the role of the *input device* that collects signals from other neurons and transmits them to the soma. The soma is the *central processing unit* that performs an important non-linear

processing step: if the total input exceeds a certain threshold, then an output signal is generated. The output signal is taken over by the *output device*, the axon, which delivers the signal (spike train) to other neurons.

Since all spikes of a given neuron look alike, the form of the action potential does not carry any information. Rather, it is the number and the timing of spikes, which matter.

In this paper we adopt the Izhikevich model

$$\begin{array}{ll} C\dot{v} = k\left(v - v_r\right)\left(v - v_t\right) - u + I & \text{if} v \geq v_{peak} \text{then} \\ \dot{u} = a\left\{b\left(v - v_r\right) - u\right\} & v \leftarrow c, u \leftarrow u + d \end{array} \tag{1}$$

which has only 9 dimensionless parameters. Depending on the values of a and b, it can be an integrator or a resonator. The parameters c and d take into account the action of high-threshold voltage-gated currents activated during the spike, and affect only the after-spike transient behavior. v is the membrane potential, u is the recovery current, C is the membrane capacitance, v_r is the resting membrane potential, and v_t is the instantaneous threshold potential. The parameters k and b can be found when one knows the neuron's rheobase and input resistance. The sign of b determines whether u is an amplifying ($b < 0$) or a resonant ($b > 0$) variable. The recovery time constant is a. The spike cutoff value is v_{peak} , and the voltage reset value is c. The parameter d describes the total amount of outward minus inward currents activated during the spike and affecting the after-spike behavior. A detail description of the model can be found in [6,7].

3 Proposed Method

It is important to point out that when the input current signal changes the response of the spiking neuron also changes, generating different firing rates. The firing rate is computed as the number of spikes generated in an interval of duration T divided by T. The neuron is stimulated during T ms with an input signal and fires when its membrane potential reaches a specific value generating an action potential (spike) or a train of spikes.

The proposed method is inspired in the method described in [19]. We expect that patterns which belong to the same class generate similar firing rates and patterns, which belong to other classes generate firing rates different enough to discriminate among the classes.

Let $\mathrm{D}=\left\{\mathbf{x}^i, k\right\}_{i=1}^{p}$ be a set of p input patterns where $k = 1, \ldots, K$ is the class to which $\mathbf{x}^i \in \mathbb{R}^n$ belongs. First of all, each input pattern is transformed into an input signal I. The spiking neuron model is not directly stimulated with the input pattern $\mathbf{x}^i \in \mathbb{R}^n$, but with an injection current I. Since synaptic weights of the model are directly connected to the input pattern $\mathbf{x}^i \in \mathbb{R}^n$, the injection current generated with this input pattern can be computed as

$$I = \gamma \cdot \mathbf{x} \cdot \mathbf{w} \tag{2}$$

where $\mathbf{w}^i \in \mathbb{R}^n$ is the set of synaptic weights of the neuron model and $\gamma = 100$ is a gaining factor which guarantees that the neuron will fire.

After that, the spiking neuron is stimulated using I during T ms and then the firing rate of the neuron is computed. With this information, the average firing rate $\mathbf{AFR} \in \mathbb{R}^K$ of each class is computed.

At last, the class of an input pattern $\tilde{\mathbf{x}}$ is determined by means of the firing rates as follows:

$$cl = \arg\min_{k=1}^{K} (|AFR_k - fr|) \tag{3}$$

where fr is the firing rate generated by the neuron model stimulated with the input pattern $\tilde{\mathbf{x}}$.

The synaptic weights of the model, which are directly connected to the input pattern, determines the firing rate of the neurons. This means that our learning phases consist in generating the desired behavior by adjusting the synaptic weights of the neuron. The learning phase will be done by the particle swarm optimization algorithm.

4 Adjusting Synapses of the Neuron Model

Synapses of the neuron model $\boldsymbol{w}$ are adjusted by means of the PSO algorithm. As we already said, the PSO algorithm is a powerful and an efficient technique for optimizing non-linear and non-differentiable continuous space functions. This algorithm works with a population of particles $\mathbf{x}_i$ that represent solutions (positions). Each particle changes its position with the time t. They are guided by the knowledge of the best particle from whole population and also by its own knowledge as so to search the optimums value. In this case, each one is guided by a velocity function that evolves a social component $\mathbf{p}_g$ (the best particle of all) and a cognitive component $\mathbf{p}_i$ (the best position of each particle). The velocity function limited to the range $[v_{min}, v_{max}]$ help to find the best positions in the search space. This function is shown in the next equation:

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 r_1 (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2 r_2 (\mathbf{p}_g(t) - \mathbf{x}_i(t)) \tag{4}$$

where ω is the inertia weight $[1, 0)$ that changes each iteration, c_1 and c_2 are acceleration coefficients; $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$ are uniformly distributed random numbers called the craziness After that, each particle i has to update its new position computing by the next equation:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{5}$$

Moreover, each possible solution is measured by means of fitness function in order to know its aptitude. It is desired that this solution be the optimums or the nearest to it.

Finally, the PSO algorithm for the real version could be performed as follows:

Given a population of $\mathbf{x}_i \in \mathbb{R}^D, i = 1, \ldots, M$ individuals

1. Initialize the population at random.
2. Until a stop criteria is reached:
   (a) For each individual $\mathbf{x}_i$ evaluate their fitness

(b) For each individual i, update its best position $\mathbf{p}_i$.
(c) From all individual i, update the best individual $\mathbf{p}_g$.
(d) For each individual i, compute the velocity update equation $\mathbf{v}_i(t+1)$ and then compute the current position $\mathbf{x}_i(t+1)$.

To find the synaptic weights that maximize the accuracy of the spiking neural model during a pattern recognition task, the next fitness function was proposed:

$$f(\boldsymbol{w}, D) = 1 - performance(\boldsymbol{w}, D) \tag{6}$$

where $\boldsymbol{w}$ are the synapses of the model, D is the set of input patterns and $performance(\boldsymbol{w}, D)$ is a function which follows the steps described in above section and computes the classification rate given by the number of patterns correctly classified divided by the number of tested patterns.

5 Experimental Results

Several experiments were performed in order to evaluate the accuracy of the proposed method. Five of them were taken from the UCI machine learning repository [8]: iris plant, glass, diabetes, liver-bupa and wine datasets. In addition, another dataset was generated from a real object recognition problem [9].

The parameters for the Izhikevich neuron were defined as $C = 100$, $v_r = -60$, $v_t = -40$, $v_{peak} = 35$, $k = 0.7$, $a = 0.03$, $b = -2$, $c = -50$, and $d = 100$. The Euler method was used to solve the differential equation of the model with $dt = 1$. The parameter to compute input current I from the input pattern was set to $\theta = 100$ with a duration of $T = 1000$. For the case of the particle swarm optimization algorithm, $NP = 40$, $MAXGEN = 1000$, $VMAX = 4$, $VMIN = -4$, $c_1 = 2$, $c_2 = 2$, $XMAX = 10$, $XMIN = -10$ and ω was varied in the range $[0.95 - 0.4]$ throught the generations.

The accuracy (classification rate) achieved with the proposed method was computed as the number of input patterns correctly classified divided by the total number of tested patterns. To validate the accuracy of the method 10 experiments over each dataset were performed. Something important to notice is

Table 1. Average accuracy provided by the methods using different databases

Dataset	Method using DE [19]		Proposed method using PSO	
	Tr. cr.	Te. cr.	Tr. cr.	Te. cr.
Wine	0.9796	0.8744	0.9782	0.8879
Iris plant	0.9933	0.9833	0.9933	0.97
Glass	0.8158	0.7411	0.8178	0.7457
Diabetes	0.8038	0.7371	0.7990	0.7619
Liver	0.7620	0.6870	0.7591	0.6754
Object rec.	1	0.9850	1	0.9950

Tr. cr = Training classification rate, Te. cr. = Testing classification rate.

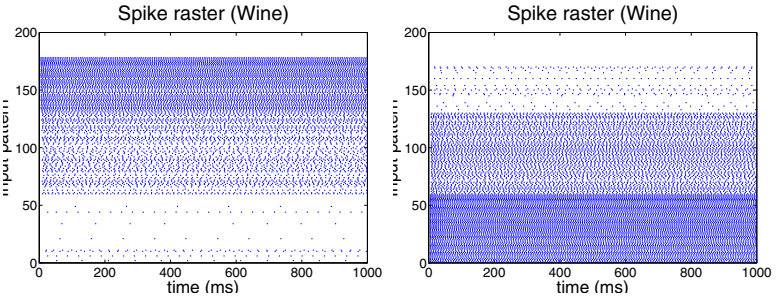


Fig. 1. Two of the experimental results obtained with the wine dataset. Notice that 3 different firing rates which correspond to 3 classes can be observed.

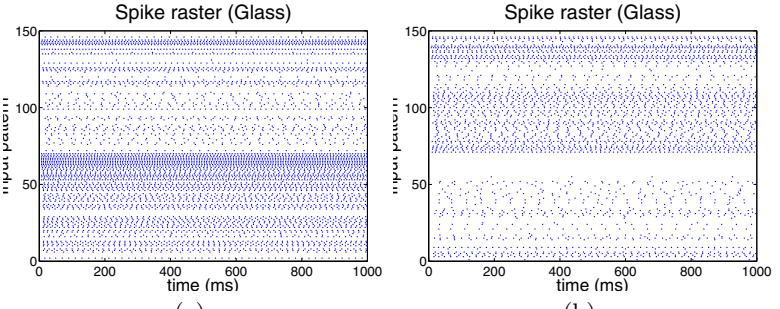


Fig. 2. Two of the experimental results obtained with the glass dataset. Notice that 2 different firing rates which correspond to 2 classes can be observed.

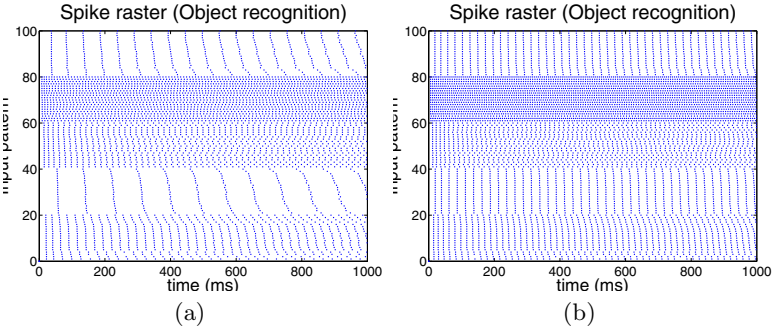


Fig. 3. Two of the experimental results obtained with the object recognition dataset. Notice that 5 different firing rates which correspond to 5 classes can be observed.

that in each experiment the datasets were randomly split into two new partitions: 50% of the patterns for training and the remain for testing.

Due to space only some of the experimental results achieved with the proposed method are shown in Fig 1 - Fig 3. As can be appreciated from these figures, the set of synaptic weights found with the PSO algorithm provokes that the spiking neuron generates almost the same firing rate when it is stimulated with input patterns from the same class. On the contrary, the spiking neuron generates firing rates different enough to discriminate among the different classes when it is stimulated with input patterns which belong to different classes.

The average classification rate computed from all experimental results is shown in Table 1. These preliminary results suggest that spiking neurons trained with a PSO algorithm can be considered as an alternative way to perform different pattern recognition tasks. As the reader can appreciate, only one spiking neuron was enough to solve a pattern recognition problem with a high acceptable accuracy. In general, the accuary of the proposed method was comparable to the method described in [19]. Both methods provide similar results; however, in some problems the proposed method was slightly better than the other, and vice verse.

We can also conjecture that if only one neuron is capable of solving pattern recognition problems, perhaps several spiking neurons working together can improve the experimental results obtained in this research. However, that is something that should be proven.

6 Conclusions

In this paper a new method to apply a spiking neuron in a pattern recognition task was proposed. This method is based on the firing rates produced with a spiking neuron when is stimulated. The training phase of the neuron model was done by means of a particle swarm optimization algorithm. After training, we observed that input patterns, which belong to the same class, generate almost the same firing rates and input patterns, which belong to different classes, generate firing rates different enough to be discriminate among the different classes.

Through several experiments, we can conclude that spiking neurons can be considered as an alternative tool to solve pattern recognition problems. Concerning to the strategy adopted to adjust the synaptic weights, we could observe that the results achieved with the particle swarm optimization algorithm are comparable to those provided by the differential evolution algorithm.

Nowadays, we are developing a methodology to calculate the maximum number of categories that the spiking neuron can classify. Furthermore, we are researching different alternatives of combining several types of spiking neurons in one network to improve the results obtained in this research and then apply it in more complex pattern recognition problems such as face, voice and 3D object recognition.

Acknowledgements

The author thanks Universidad La Salle for the economical support under grant number ULSA I-113/10.

References

1. Garro, B.A., Sossa, H., Vazquez, R.A.: Design of Artificial Neural Networks using a Modified Particle Swarm Optimization Algorithm. IJCNN, 938–945 (2009)
2. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Networks 10(9), 1659–1671 (1997)
3. Loiselle, S., Rouat, J., Pressnitzer, D., Thorpe, S.: Exploration of rank order coding with spiking neural networks for speech recognition. IJCNN 4, 2076–2080 (2005)
4. Thorpe, S.J., Guyonneau, R., et al.: SpikeNet: Real-time visual processing with one spike per neuron. Neurocomputing 58(60), 857–864 (2004)
5. Di Paolo, E.A.: Spike-timing dependent plasticity for evolved robots. Adaptive Behavior 10(3), 243–263 (2002)
6. Izhikevich, E.M.: Simple model of spiking neurons. IEEE Trans. on Neural Networks 14(6), 1569–1572 (2003)
7. Izhikevich, E.M.: Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. The MIT press, Cambridge (2007)
8. Murphy, P.M., Aha, D.W.: UCI repository of machine learning databases. Dept. Inf. Comput. Sci., Univ. California, Irvine (1994)
9. Vazquez, R.A., Sossa, H.: A new associative model with dynamical synapses. Neural Processing Letters 28(3), 189–207 (2008)
10. Vazquez, R.A., Cachon, A.: Integrate and fire neurons and their application in pattern recognition. In: Proceedings of the 7th CCE, pp. 424–428 (2010)
11. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948 (1995)
12. Zhao, L., Yang, Y.: PSO-Based Single Multiplicative Neuron Model for Time Series Prediction. Expert Systems with Applications 36, 2805–2812 (2009)
13. Yu, J., et al.: An Improved Particle Swarm Optimization for Evolving Feedforward Artificial Neural Networks. Neural Processing Letters 26, 217–231 (2007)
14. Gudise, V.G., Venayagamoorthy, G.K.: Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 110–117 (2003)
15. Vázquez, R.A.: Pattern recognition using spiking neurons and firing rates. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 423–432. Springer, Heidelberg (2010)
16. Hamed, H.N., Kasabov, N., Michlovský, Z., Shamsuddin, S.M.: String Pattern Recognition Using Evolving Spiking Neural Networks and Quantum Inspired Particle Swarm Optimization. In: Leung, C.S., Lee, M., Chan, J.H. (eds.) ICONIP 2009. LNCS, vol. 5864, pp. 611–619. Springer, Heidelberg (2009)
17. Kamoi, S., et al.: Pulse Pattern Training of Spiking Neural Networks Using Improved Genetic Algorithm. In: Proceedings of the IEEE CIRA, pp. 977 – 981 (2003)
18. Hong, S., et al.: A Cooperative Method for Supervised Learning in Spiking Neural Networks. In: 14th CSCWD, pp. 22–26 (2010)
19. Vazquez, R.A.: Izhikevich Neuron Model and its Application in Pattern Recognition. Australian Journal of Intelligent Information Processing Systems 11, 35–40 (2010)

Clustering Aggregation for Improving Ant Based Clustering

Akil Elkamel[1,2], Mariem Gzara[1,2], and Hanêne Ben-Abdallah[1,3]

[1] MIRACL (Multimedia InforRmation systems and Advanced Computing Laboratory) Route de Tunis Km 10, B.P. 1030, Sfax, 3021 Tunisia
[2] Institut Supérieur d'Informatique et de Mathématiques de Monastir Avenue de la Korniche - B.P. 223 - Monastir - 5000, Tunisia
[3] Faculté des Sciences Économiques et de Gestion de Sfax, Route de l'aéroport Km 4 Sfax, 3018, Tunisia
akil.elkamel@gmail.com, mariem.gzara@isimsf.rnu.tn, hanene.benabdallah@fsegs.rnu.tn

Abstract. In this paper, we propose a hybridization between an ant-based clustering algorithm: CAC (Communicating Ants for Clustering) algorithm [5] and a clustering aggregation algorithm: the Furthest algorithm [6]. The CAC algorithm takes inspiration from the sound communication properties of real ants. In this algorithm, artificial ants communicate directly with each other in order to achieve the clustering task. The Furthest algorithm takes as inputs m clusterings given by m different runs of the CAC algorithm, and tries to find a clustering that matches, as possible, all the clusterings given as inputs. This hybridization shows an improvement of the obtained results.

1 Introduction

Clustering is one of the important tasks in data mining. It has many application areas; including machine learning, biology, medicine, and statistics. The clustering problem is defined as partitioning a data set into groups (clusters) such that the data objects, in the same cluster, share a high degree of similarity while data objects, in different clusters, are much dissimilar in a remarkable way.

Referring to the literature, there are several families of methods proposed to resolve the clustering problem [7], [9], citing hierarchical, partitional, graph-based, grid-based, density-based, and model-based techniques. Hierarchical methods divide a data set into a sequence of nested partitions, while partitional algorithms divide a data set into a single partition. The graph-based clustering algorithms construct firstly a graph from the similarity matrix between the data items and then, apply a clustering algorithm to partition the constructed graph. The grid-based approaches are popular for mining clusters in a large multidimensional space wherein clusters are regarded as denser regions than their surroundings. The density-based clustering approach is a methodology that is capable of finding arbitrarily shaped clusters, where clusters are defined as dense regions separated by low-density regions. In the model-based clustering, it is assumed

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 250–259, 2011.
© Springer-Verlag Berlin Heidelberg 2011

that the data are generated by a mixture of underlying probability distributions in which each component represents a different group or cluster.

Another family of methods based on biological and natural phenomena is proposed in literature, and it's called bio-inspired methods. Bio-inspired clustering algorithms can be classified into two categories: those that start from an initial solution (or a set of solutions) and try to improve it iteratively (genetic algorithms [12], tabu search [3], simulated annealing [13] and Particle Swarm Optimization [2]) and those that use a population of artificial agents that evolve in the clustering environment to achieve the clustering task and generate a partition as a solution (Ant-based clustering algorithms [4], [11]).

We have proposed in [5] a new ant-based clustering algorithm: CAC algorithm (Communicating Ants for Clustering). The CAC algorithm takes inspiration from the acoustic communication phenomenon observed among real ants [8]. It has shown good results when it was applied on real and synthetic data sets [5]. The CAC algorithm shows also a good robustness of the obtained results despite its stochastic nature. To further improve the results, we choose to hybridize the CAC algorithm with a clustering aggregation algorithm [6] which is the Furthest algorithm, which from a given set of clusterings finds a single clustering that agrees as much as possible with the input clusterings.

This paper is organized as the following: in section 2, we describe the CAC (Communication Ants for Clustering) algorithm [5]. Section 3 is devoted to presenting the Furthest algorithm. In section 4, we introduce the main idea of this paper which is the hybridization of the CAC and the Furthest algorithm. In section 5 we explain the experimental evaluation. Section 6 concludes the paper and points out some avenues for a possible future work.

2 The CAC Algorithm

The CAC algorithm [5] is based mainly on the acoustic communication phenomenon observed in real ants [8]. In our algorithm, a set of ant-agents move on a two-dimensional, squared and toroidal grid G where the data objects are initially scattered at random. G includes $L = \lceil \sqrt{N} \rceil$ cells per side where N is the number of objects. In our approach, several objects can be placed on one cell, which form a heap of objects. In this case, a class corresponds to a heap and a partition is given by all heaps on the grid. Figure 1 illustrates this configuration.

The number of ants is automatically determined by the number of data. If L represents the side of the grid, then the number of ants $F = \frac{L^2}{9}$. The idea is that an ant is responsible for the cell where it is located and the eight nearby cells. Thus, with this formula we guarantee that the number of ants is neither very large nor very small compared to the number of cells on the grid. At the initialization, the F ants $(f_1, ..., f_F)$ are positioned randomly on the grid by checking, as for the distribution of objects, that a cell can't contain at the same time two ants or two objects. At each iteration, each ant f_i moves randomly on the grid. Thus, an ant can move to one of the eight cells of its neighborhood.

After the initialization, the ants try to find on the grid similar groups of objects in order to merge them. Each ant that reaches a cell containing an object or a

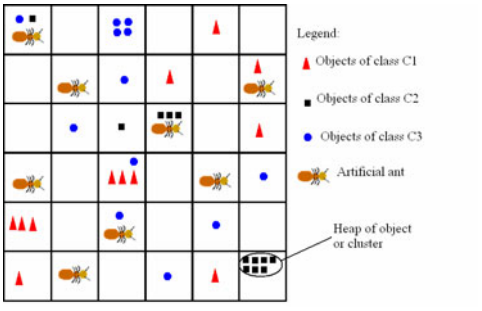


Fig. 1. CAC allows the construction of heaps of objects on the grid

heap of objects, sends a recruitment signal to other ants that hold objects or heaps of objects. This signal contains information on the characteristics of the object or the heap it holds as well as the coordinates of the cell where it is positioned on the grid. Then, each ant among those that received the signal will evaluate the similarity between its heap and the heap of the ant issuing the signal. If the similarity is over a certain threshold, it will pick up its object or its heap and drop it on the cell where the ant that sent the signal is, returns back to its initial position and continues moving by choosing, randomly, a destination among the eight neighbor cells. In this way, larger and larger clusters will be formed. After it has completed all its communication, the ant that issued the signal will move to a nearby cell seeking for other objects or heaps. Ants are considered in turn and the process is reiterated until the number of iteration T_{Max} is reached.

The similarity between two heaps of objects H_1 and H_2 is evaluated as the maximum distance between H_1 and H_2:

$$D_{max}(H_1, H_2) = \max_{x_i \in H_1, x_j \in H_2} \{d(x_i, x_j)\} \tag{1}$$

where d is a distance measure.

Two heaps H_1 and H_2 are aggregated if the maximum distance between them $D_{max}(H_1, H_2)$ is less or equal to the aggregation threshold t. This threshold is not constant, we start with a very low threshold equal to the minimum distance between all objects $d_{min}(O)$, and we add at each iteration a constant ϵ.

$$d_{min}(O) = \min_{(i,j) \in \{1,\ldots,N\}^2, i \neq j} \{d(x_i, x_j)\} \tag{2}$$

$$\epsilon = \frac{\overline{d}(O)}{T_{max}} \tag{3}$$

where $\overline{d}(O)$ is the average distance between all objects of the set O and T_{max} is the maximum number of iterations.

$$\overline{d}(O) = \frac{2}{N(N-1)} \sum_{(i,j) \in \{1,\ldots,N\}^2, i<j} d(x_i, x_j) \tag{4}$$

Algorithm 1 presents the steps of the CAC algorithm.

Algorithm 1. CAC (Communicating Ants for Clustering)

Initialization: Randomly place the data objects and the ants on the grid, by checking that two objects or two ants can not be on the same cell.
for $t = 1$ to T_{Max} **do**
  for every ant a_k, $k \in < 1, K >$ **do**
    if the ant a_k has an object [heap] **then**
      for every ant $a_i, i \in < 1, K >, i \neq k$ **do**
        Calculate the distance between his object [heap] and the object [heap] of the ant a_k.
        if this distance is accepted **then**
          move its object [heap] to the cell where the main ant a_k is located.
          a_i moves to one of the eight neighbors cells.
        end if
      end for
    end if
    a_k moves to one of the eight neighbors cells.
  end for
end for

3 The Furthest Algorithm for Clustering Aggregation

The clustering aggregation problem [6] is defined as an optimization problem where, given a set of m clusterings, we want to find the clustering that minimizes the total number of disagreements with the m clusterings.

Considering a set of n objects $V = \{v_1, ..., v_n\}$. A clustering C of V is a partition of V into k clusters $C_1, ..., C_k$. For each $v \in V$, we use $C(v)$ to denote the label of the cluster to which the object v belongs, that is, $C(v) = j$ if and only if $v \in C_j$. Considering also m clusterings: we use C_i to denote the i^{th} clustering, and k_i for the number of clusters of C_i.

A disagreement between two clustering C_1 and C_2 is a pair of object (u, v) such that C_1 places them in the same cluster, while C_2 places them in different clusters or vice versa :

$$d_{u,v}(C_1, C_2) = \begin{cases} 1 \text{ if } ((C_1(u) = C_1(v) \, and \, C_2(u) \neq C_2(v)) \\ \quad or \, (C_1(u) \neq C_1(v) \, and \, C_2(u) = C_2(v))) \\ \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

$d_V(C_1, C_2)$ denotes the number of disagreements between C_1 and C_2

$$d_V(C_1, C_2) = \sum_{(u,v) \in V \times V} d_{u,v}(C_1, C_2). \tag{6}$$

The clustering aggregation problem is formulated as follows: given a set of objects V and m clusterings $C_1, ..., C_m$ on V, compute a new clustering C that minimizes the total number of disagreements with all the given clusterings :

$$D(C) = \sum_{i=1}^{m} d(C_i, C) \tag{7}$$

In [6], the authors present many algorithms for resolving the clustering aggregation problem. The Furthest algorithm has low complexity and it shows better performances in comparison to the other studied algorithms in [6].

In [6], the clustering aggregation problem is formulated as a graph where the weight X_{uv} of the edge (u, v) is the fraction of clusterings that place u and v in different clusters. The Furthest algorithm starts by placing all the nodes into a single cluster. Then, it finds the pair of nodes that are furthest apart and places them into different clusters. These two nodes become the centers of the clusters. The remaining nodes are assigned to the center that incurs the least cost. This procedure is repeated iteratively: at each step, a new center is generated that is the furthest from the existing centers, and the nodes are assigned to the center that incurs the least cost. At the end of each step, the cost of the new solution is computed. If it is lower than that of the previous step, then the algorithm continues. Otherwise, the algorithm outputs the solution computed in the previous step.

4 Hybridization of the CAC Algorithm and the Furthest Algorithm

The CAC algorithm is an ant-based clustering algorithm inspired by the acoustic communication phenomenon observed in real ants. The algorithm was tested and evaluated on several real data sets.

The CAC algorithm generated good results, it showed a superiority over the k-means and other ant-based clustering algorithms on several well known benchmark databases. The CAC algorithm is a stochastic algorithm. In fact, objects and ants are initially randomly scattered on the grid. Moreover, ants move randomly in their neighborhood throughout the algorithm. Despite this stochastic nature the CAC algorithm showed a good robustness. If we run the CAC algorithm several times we have, in average, good results and the standard deviation of all the evaluation measures is very low. But when comparing two different runs, we can find disagreements between the generated partitions. These disagreements are caused by the stochastic steps of the algorithm. That's why we thought to hybridize the CAC algorithm with the Furthest algorithm to have a final clustering that matches as possible with the clusterings in result of the CAC algorithm. The hybridization consists on running the CAC algorithm m times and the Furthest algorithm takes the resulted m clusterings as inputs. Then, it generates a clustering that agrees as much as possible with the clusterings generated by the ants.

5 Experimental Evaluation

5.1 Data Sets

To achieve our experimental evaluation, we have used real data sets issued from the machine learning repository [1] and synthetic data sets (table 1). These data sets are supervised (we know the class of each object) in order to assess the quality of the partitioning we get. We present for each data set the dimension of objects space (M), the number of classes (K) and the total number of objects (N).

Table 1. Data sets used for the experimental evaluation

Data sets	**M**	**K**	**N**
Iris	4	3	150
Wine	12	3	178
Glass	9	7	214
Thyroid	5	3	215
Breast cancer wisconsin	10	2	699
Yeast	8	10	1484
Synth1	2	4	2000

5.2 Evaluation Measures

F-measure: The F-measure is based on the idea of comparing a resulting partition with a real or a reference partition. It uses the recall and precision measures which are defined as follows:

$$recall(i,j) = \frac{n_{ij}}{N_i} \tag{8}$$

$$precision(i,j) = \frac{n_{ij}}{N_j} \tag{9}$$

where n_{ij} is the number of objects or individuals present in the reference class C_i and in the resulting class C_j. N_i and N_j represent respectively the total number of objects in the class C_i and C_j.

Or $recall(i,j)$ and $precision(i,j)$ are relative only for one case, that of the adequacy of reference class C_i with the result class C_j. To evaluate the entire class C_i, we just choose the maximum of the recall and precision values obtained within C_i:

$$recall(i) = \max_j [recall(i,j)] \tag{10}$$

$$precision(i) = \max_j [precision(i,j)] \tag{11}$$

The global values of the recall and precision for all classes will be finally determined as follows:

$$recall = \sum_i p_i \times recall(i) \tag{12}$$

$$precision = \sum_i p_i \times precision(i) \tag{13}$$

where p_i is the weight of the class C_i. It is given by the following formula:

$$p_i = \frac{N_i}{\sum_k N_k} \tag{14}$$

The global value of the F-measure will be given by the following formula:

$$F = \frac{2 \times precision \times recall}{precision + recall} \tag{15}$$

Note that high values of recall, precision and F-measure matches to a best clustering.

Impurity index: If a clustering contains k clusters with sizes $s_1, ..., s_k$, and the sizes of the majority class in each cluster are $m_1, ..., m_k$, respectively, then the impurity index measure is defined as:

$$I = \sum_{i=1}^{k} (\frac{s_i - m_i}{s_i}) \tag{16}$$

If a clustering has I value equal to 0, it means that it contains only pure clusters.

Rand index: Given a set of n objects $O = \{o_1, o_2, ..., o_n\}$ and two partition $P = \{C_1, C_2, ..., C_k\}$ and $P' = \{C'_1, C'_2, ..., C'_{k'}\}$, respectively containing k and k' clusters. We define the following measures:

- a, the number of pairs of elements in O, that are in the same cluster in P and in the same cluster in P'.
- b, the number of pairs of elements in O, that are in different clusters in P and in different clusters in P'.
- c, the number of pairs of elements in O, that are in the same cluster in P and in different clusters in P'.
- d, the number of pairs of elements in O, that are in different clusters in P and in the same cluster in P'.

The Rand index is calculated as follow:

$$R = \frac{a + b}{a + b + c + d} = \frac{a + b}{\binom{n}{2}} \tag{17}$$

A best clustering is the one that maximizes the Rand index value.

5.3 Experimentations

In our experimentation study, we ran the CAC algorithm 5 times and we have provided the resulting 5 clusterings to the Furthest algorithm.

The *Synth1* data set is the type of data most frequently used within ant-based clustering algorithms. It is two-dimensional and consists of four clusters. They are generated by four different Normal Distributions and each has a size of 500. Figure 2 shows the partitions generated by 5 runs of the CAC algorithm and that is generated by the Furthest algorithm after having as inputs the 5 clusterings $C_1, ..., C_5$. Figure 2 shows that the Furthest algorithm succeed to correct misclassifications generated by the ants in each clustering.

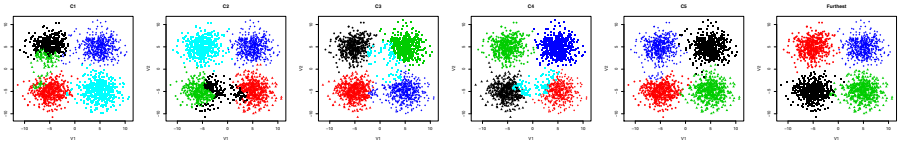


Fig. 2. Results on *Synth1* data set

We present in the following tables the results on the different data sets. For each clustering within the m input clusterings and the final clustering, we present the number of resulted clusters (#clu), the recall (R), the precision (P), the Fmeasure (F), the impurity index (I) and the rand index ($Rand$). The average of all the evaluation measures is represented by the $Avg(C_{1-5})$ row.

Table 2 shows that the clustering obtained by the hybridization of the CAC algorithm and the Furthest algorithm for the iris and wine data sets is better than the average of the five clusterings.

Table 3 shows that for the glass data set, the clustering C has recorded a slight superiority for the different evaluation measures, except the recall measure. This exception can be justified by the number of the obtained clusters which is larger than the average number of clusters of the 5 input clusterings. For the thyroid data set, the Furthest algorithm showed an improvement of the final clustering among all the evaluation measures.

Table 4 presents the results for the Breast cancer wisconsin and the Yeast data sets. For the Breast cancer wisconsin, the aggregation of the five clusterings has recorded better results compared to the average of all the evaluation measures, while the clustering C_4 performs better than the clustering C. For the Yeast data set the Furthest algorithm generates a larger number of clusters (14) compared to the real number of clusters (10), but when looking at the precision value, we notice that the clusters generated by the Furthest algorithm are more homogeneous than those generated by the five clusterings given as inputs.

We have presented in tables 2, 3 and 4 the evaluation measures values of the clustering obtained by the Furthest algorithm after having as inputs five clusterings generated by five runs of the CAC algorithm. The Furthest algorithm always generates a clustering with a better precision than the inputs clusterings. It succeeds to correct the misclassifications committed by the ants.

Table 2. Results on the *Iris* and *Wine* data sets

	Iris						*Wine*					
Clustering	**#clu**	R	P	F	I	$Rand$	**#clu**	R	P	F	I	$Rand$
C_1	3	0.96	0.96	0.96	0.04	0.94	4	0.83	0.95	0.89	0.14	0.81
C_2	4	0.80	0.98	0.88	0.04	0.90	4	0.81	0.89	0.85	0.14	0.82
C_3	3	0.96	0.96	0.96	0.03	0.95	4	0.88	0.96	0.92	0.07	0.87
C_4	4	0.73	1.00	0.84	0.10	0.85	3	0.79	0.68	0.73	0.37	0.68
C_5	3	0.95	0.95	0.95	0.04	0.94	4	0.88	0.97	0.92	0.08	0.87
$Avg(C_{1-5})$	3.4	0.88	0.97	0.91	0.05	0.91	3.8	0.83	0.89	0.86	0.16	0.81
C	3	0.96	0.96	0.96	0.04	0.95	4	0.90	0.97	0.93	0.06	0.89

Table 3. Results on the *Glass* and *Thyroid* data sets

	Glass						*Thyroid*					
Clustering	**#clu**	R	P	F	I	$Rand$	**#clu**	R	P	F	I	$Rand$
C_1	6	0.88	0.51	0.64	0.52	0.61	6	0.83	0.91	0.87	0.10	0.82
C_2	7	0.75	0.71	0.73	0.50	0.63	6	0.83	0.91	0.87	0.09	0.82
C_3	8	0.75	0.72	0.73	0.50	0.63	6	0.83	0.91	0.87	0.09	0.83
C_4	6	0.83	0.70	0.76	0.52	0.61	6	0.65	0.94	0.77	0.13	0.68
C_5	7	0.77	0.56	0.65	0.51	0.59	7	0.80	0.96	0.88	0.07	0.86
$Avg(C_{1-5})$	6.8	0.80	0.64	0.70	0.51	0.62	6.2	0.78	0.92	0.85	0.096	0.80
C	8	0.75	0.72	0.74	0.50	0.63	6	0.83	0.92	0.87	0.09	0.83

Table 4. Results on the *Breast cancer wisconsin* and *Yeast* data sets

	Breast cancer wisconsin						*Yeast*					
Clustering	**#clu**	R	P	F	I	$Rand$	**#clu**	R	P	F	I	$Rand$
C_1	2	0.90	0.91	0.91	0.09	0.83	11	0.61	0.70	0.65	0.59	0.64
C_2	2	0.89	0.90	0.89	0.10	0.81	11	0.49	0.71	0.58	0.58	0.62
C_3	2	0.85	0.88	0.86	0.14	0.75	10	0.74	0.77	0.75	0.59	0.58
C_4	2	0.91	0.92	0.92	0.08	0.84	10	0.76	0.69	0.72	0.61	0.54
C_5	2	0.90	0.91	0.90	0.09	0.82	11	0.50	0.80	0.61	0.59	0.62
$Avg(C_{1-5})$	2	0.89	0.90	0.89	0.10	0.81	10.6	0.62	0.73	0.66	0.59	0.60
C	2	0.90	0.91	0.91	0.09	0.83	14	0.46	0.85	0.60	0.58	0.65

6 Conclusion

In this paper, we have presented a hybridization of the CAC algorithm with the Furthest algorithm. This hybridization shows an improvement of the obtained results. As prospects, we attempt to adapt the principles developed in CAC and its hybridization with the Furthest algorithm to handling large data sets.

References

1. Blake, C., Merz C.: UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences
2. Cui, X., Potok, T.E., Palathingal, P.: Document Clustering using Particle Swarm Optimization. In: IEEE Swarm Intelligence Symposium, The Westin (2005)
3. Delgado, M., Skrmeta, A.G., Barber, H.M.: A Tabu Search Approach To The Fuzzy Clustering Problem. In: Proceedings of the Sixth IEEE International Conference on Fuzzy Systems, pp. 125–130 (1997)
4. Deneubourg, J.L., Goss, S., Franks, N., Sendova Franks, A., Detrain, C., Chretien, L.: The dynamics of collective sorting: robot-like ant and ant-like robots. In: Meyer, J.-J., Wilson, S. (eds.) Proceedings of the First International Conference on Simulation of Adaptative Behavior, Paris, France, pp. 356–365 (1990)
5. Elkamel, A., Gzara, M., Jamoussi, S., Ben Abdallah, H.: An ant-based algorithm for clustering. In: The 7th ACS/IEEE International Conference on Computer Systems and Applications, Rabat, Morocco, pp. 76-82 (2009)
6. Gionis, A., Mannila, H., Tsaparas, P.: Clustering Aggregation. ACM Transactions on Knowledge Discovery from Data 1(1), Article 4 (2007)
7. Guojun, G., Chaoqun, M., Jianhong, W.: Data Clustering: Theory, Algorithms, and Applications. ASA-SIAM Series on Statistics and Applied Probability (2007)
8. Hickling, R., Brown, R.L.: Analysis of acoustic communication by ants. J. Acoust. Soc. Am. 108(4), 1920–1929 (2000)
9. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31(3), 264–323 (1999)
10. Knight, T., Timmis, J.: A Multi-Layered Immune Inspired Approach to Data Mining. In: Lotfi, A., Garibaldi, J., John, R. (eds.) Proceedings of the 4th International Conference on Recent Advances in Soft Computing, Nottingham, UK, pp. 266–271 (December 2002)
11. Lumer, E.D., Faieta, B.: Diversity and adaptation in populations of clustering ants. In: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, pp. 501–508 (1994)
12. Raghavan, V.V., Birchard, K.: A clustering strategy based on a formalism of the reproductive process in natural systems. In: Proceedings of the Second International Conference on Information Storage and Retrieval, pp. 10–22. ACM, New York (1979)
13. Selim, S.Z., Alsultan, K.: A simulated annealing algorithm for the clustering problem. Pattern Recognition 24(7), 1003–1008 (1991)

Multi-cellular-ant Algorithm for Large Scale Capacity Vehicle Route Problem

Jie Li, Yi Chai, Penghua Li, and Hongpeng Yin

Automation College of Chongqing University,
Room 1911, Main building, Section A of Chongqing University, Chongqing, China
leighby16@gmail.com

Abstract. This paper presents a multi-cellular-ant algorithm for large scale capacitated vehicle routing problem with restrictive vehicle capability. The problem is divided into corresponding smaller ones by a decomposition methodology. Relative relationship between subsystems will be solved through cooperative performance among cellular ants to avoid trivial solutions. The empirical results composed with adaptive ant colony algorithm and traditional collaboration show more efficiency and availability.

Keywords: multi-cellular-ant algorithm, cooperation, decomposition, capacitated vehicle routing problem.

1 Introduction

Vehicle route problem (VRP) is designing delivery routes to meet some requirements and obtain minimal total cost synchronously. capacitated vehicle routing problem (CVRP) is an extension of VRP where vehicle capability is restrained. Researches on CVRP focus on two positions. On the one hand, average distance minimization is to diminish distribution cost. Ellipse rule approach made high decline by about 44% (*Santos L et al*, 2009). Heuristic algorithms played signature performances on CVRP under certain problem sizes, such as evolutionary algorithm(*Borgulya I*, 2008), genetic algorithm (*Chung − Ho Wang et al*, 2010) and particle swarm optimization (*Ai TJ et al*, 2009). (*Istvan Borgulya*, 2008) proposed a multi-objective algorithm based on extended virtual loser for CVRP as similar as other evolutionary algorithm.

(*Jens Lysgaard et al*, 2004) presented a new branch-and-cut algorithm including several separation methods to solve the three instances and find out their upper bounds and then (*Ricardo Fukasawa et al*, 2006) improved it into Branch-and-Cut-and-Price method to gain lower bounds. On the other hand, CVRP scale caught some attention. (*Jari Kytojokia et al*, 2007) scattered customer services location with known demands geographically to breakdown large scale problem. Furthermore, (*David Mestera et al*, 2007) enhanced the guided local search and evolution strategies meta-heuristics. A deoxyribonucleic acid computing and modified *Adleman − Lipton* model accelerates the search on large nodes CVRP in a decentralized model (*Yuvraj*, 2009).

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 260–266, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Subsystems coupling is ignored generally in a decentralized model. Multi-ant algorithm is proposed in a distributed model where iteration between subsystems is taken into account. Large scale CVRP is decomposed into a set of smaller ones with connection based on Tree Cut-Set algorithm given in the paper. Moreover, additional reward function punishes suboptimal strategies at each state. (Carlos Bermudez et al,2010) had proposed recombination operator for genetic search and provided solution for CVRP based on cellular model where cellular ants played well. Consequently, complex and abundant macroscopic phenomenon in the parallel evolutionary is described as cellular automata (CA) as a discrete grid dynamics model both in timespace and state vector.

The paper is organized as follows: Section 2 will introduce decomposition of large scale CVRP using tree cut-set. Section 3 describes multi-cellular-ant algorithm in a distributed model established in Session 2 based on reward shaping. The distributed multi-cellular-ant in a grid net of subsystems updates information by local rules following same action regulations synchronously with finite discrete states. Simultaneously, the procedure of cooperation between subsystems is gotten. Simulations are executed among three algorithms in Section 4. Finally, discussion and conclusions are in Section 5.

2 Decomposition for Large Scale CVRP

Large scale CVRP is deemed as a weighted incomplete undirected graph divided into subsystems by Tree Cut-Set (TCS). For the complexity, undirected graph with unsteady capacity constraints is transformed into Tree Description ($Chen\ Yulin\ et\ al, 2002$) firstly using network ripping ($Li\ Yan\ et\ al, 2004$). Then, the semantic representation of CVRP could be replaced by subsystems. The amount of subsystems is related with the carrying capability of vehicles, demands and association of customers.

We will explain $t-sepset$ couple based on $d-sepset(Ng.A\ et\ al, 1999)$.

Definition 1. In a tree, a tree-node i has and only has two different parents, has and only has one child node j, at the same time, node j has and only has two different children. Then, (i, j) is called as a t-sepset couple nodes.

Formed tree is searching by order under Greedy policy and ripping by t-sepset couple nodes of customers with stochastic demands. As what Bayesian Theory [11] says, demands are weighted by w_i. A random variable must fix some preconditions: $w > 0$ and $\sum_{i=1}^{2} w_i = 1$. Summation of demands t_a in subsystems must be lower than b limited vehicle capability. Error $e = t_a - b$ declines as lower as possible. Our procedure stops till the value of e is lower than a by constant p. Otherwise, TCS will continue.

3 Multi-cellular-ant Algorithm

A subsystem is described as a five-tuple discount value MDP model $M = \{S, A, T, R, fv\}$ where $S = \{s\}$ is a set of states, $A = \{a\}$ is a set of actions,

$T = \{p(\cdot|s,a), s \in S, a \in A\}$ are the transition probability distribution of next-state. $p(\cdot \mid s,a)$ represents the probability of action a. $R(s,a,s')$ describes reward function and fv defines additional reward function. $\pi(s)$ is policy function in the state space.

$R'(s,a,s') = R(s,a,s') + \Phi(s') - \Phi(s)$ is for any policy $\pi(s)$ and $V^*(s) = \pi(s) - \Phi(s)$ [12] by assumption that potential function $\Phi(s)$ exists. If reward function R of model M exchanges into new reward function R' of model M', reward shaping incurs optimal policies of model M being as well as optimal policies of model M'. $fv(s,a,s')$ carries ant colony information, thus R' is defined as follows :

$$R'(s,a,s') = R(s,a,s') + fv(s,a,s') \tag{1}$$

3.1 Cellular Ants

Cellular automata (CA) depicts a discrete model with a finite number of states. Lattice cells work in communication, computation, construction, growth, reproduction and evolution. Simultaneously, CA masters at ordering, turbulence, chaos, symmetry-breaking and fractals in the dynamic system. Cell state at next time is dominated by current state and neighbors' current states.

CAs structure consists of four parts: cellular-ant space, grid dynamics net, local principles and transition function. Cellular Space in subsystems has two-dimensional uncertainty states. Updated function with cellular ants information at time t under extended Moore neighbor model is as: $f : s_{t+1}^i = f(s_t^i, s_t^N)$.

The key part of *CAs* is to establish corresponding-neighbors policies under extended Moore neighbor model where last-state performances are compared with neighbors'. Thus reward function of distributed multi-cellular-ant algorithm is written as $f = f(s_t^i(a), \sum_{j=0}^{N} s_t^j(a))$ and $fv(s,a,s') = F(s_{t+1}^i)$:

$$R'(s,a,s') = R(s,a,s') + \sum_{r=0}^{2r} f(s_t^{i+r}(a), \sum_{j=0}^{N} s_t^j(a)) \tag{2}$$

3.2 Distributed Multi-cellular-ant Algorithm

Dynamic learning in Back Propagation ($Xiao - Hu\ Yu\ et\ al, 1995$) accelerates the learning rate that represented by $\varphi_t(i)$(the value of ant i at time t). $\varphi_t(i)$ value reduces to zero gradually in the limited search procedure. Let $\varphi_t(i) > 0$ be a series of constants for every ant at time t and fit the equation: $\lim_{T \longrightarrow \infty} \sum_{t+1}^{T} \varphi_t(i) = \infty$.

The influence of perturbation from unstable situations can be amended by Performance Potential ($CaoXi - enetal, 1997$). Such as, vehicle diversity may impact on the efficiency and the weather will delay the arriving time of transportation. The last-step state is chosen for performance potential for current-step state. Let $X = \{X_t, t = 1, 2 \dots\}$ picture the decision progress. According

to reward function, while s is the last-step state, the description of performance potential under state s' is defined as:

$$g_{s'} = \lim_{T \longrightarrow \infty} \{E[\rho^N \sum_{s=0}^{N-1} R'(s,a,s')|X_0 = s] \\ -(N-1)g_s\} \tag{3}$$

Reward shaping integrates with ant colony algorithm based on Q-learning (*Bagnell, J.etal*, 2006; *Diettericheta l*, 2000):

$$\pi_k^*(i,j) = argminmax\{fv(s,a,s') \sum_{a \in A} \pi_a(i,j) * Q^\star\} \tag{4}$$

$$Q^\star = max_{a \neq k} Q(i,j,k) - Q(i,j,s) \tag{5}$$

$$Q(i,j,k) = \varphi_t(i) \cdot Q(1 - r_k/R_k) - \delta \cdot g_{s'} \\ +r_k/R_k \cdot V^\star \tag{6}$$

$$V^\star = R'(i,k,j) + \gamma V^*(s') \tag{7}$$

Where α is discounted factor. While the amount of cellular ants arriving at city i is R_k, the amount of cellular ants chosen city j as the next city is r_k.

4 Experimental Result

The dataset for simulation is from a GIS software, "ArcView", transforming geographic location in the original city map to network one. Information in ArcView compose latitude and longitude vector. The amount of cellular ants is 31. Parameter α is trail evaporation coefficient. The revisited probability will increase if α is high. Parameter β also takes heuristic information where best regions are $0.1 \leq \alpha \leq 0.3$ and $3 \leq \beta \leq 6$ (*JIANG Ling − yan et al*, 2007). Then parameters in Q-learning are set as: $\alpha = 0.6$, $\beta = 4$, $\gamma = 0.8$, $Q = 2$ and $\rho = 0.7$.

4.1 Solution Superiority

The number of places in the city is 500. Time consumes ordered by adaptive ant colonycooperation without decomposition and cooperation with decomposition are 1.7514e+003 | 202.8906 | 179.5156. The shortest distances by the same orders are 1.4263e+004 | 1.4164e+003 | 1.3541e+003. For larger scale problem, multi-cellular-ant cooperation with decomposition displays quicker astringency than cooperation without decomposition. Moreover, solutions through multi-cellular-ant algorithm are better (illustrated by the right one in Fig 1). Without decomposition, the cooperation is easy to trap into local solution (illustrated by the left one in Fig 1).

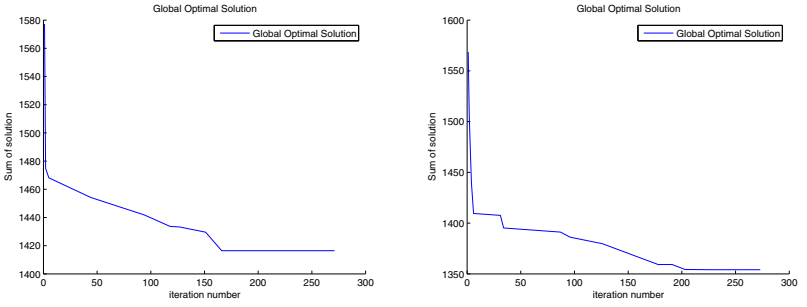


Fig. 1. Shortest distances comparison

4.2 Efficiency for System Coupling

For the large scale problem, interactions between subsystems drive signature impacts on operations. Interaction and coordination are special characters of multi-cellular-ant algorithm with corresponding coupling. With various scales of CVRP, performance contrasts are analyzed during experiments.

From first two graphs in Fig 2, best solutions and average solutions under multi-cellular-ant algorithm and multi-ant algorithm without decomposition are shown clearly. While problem is simple, accomplishments between those two methods differentiate infirmly. The later method leads to suboptimal operations rapidly with incremental problem scopes.

Because of restrained vehicle capacity, cellular ants keep contact with each others through relative reinforcement learning. Therefore, more complex manipulations of multi-cellular-ant algorithm takes more time for smaller problem. However, under fewer iterations, it costs less time to find out optimal solutions as higher scale of CVRP (illustrated by the third graph in Fig 2).

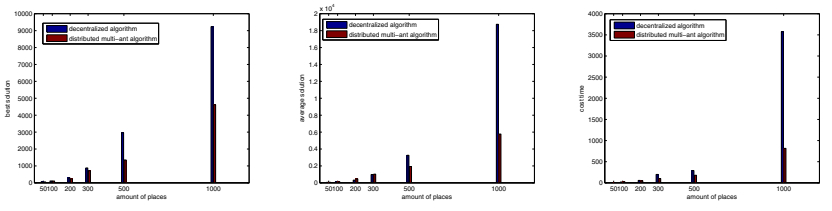


Fig. 2. Results comparisons

Performance Potential(Pp) is helpful for filtering system noises. The purpose of importing Pp into multi-cellular-ant algorithm is to reduce impression from uncertainties and accelerate actions. Its performances is determined by parameter δ. Simulations are executed by several δ values (illustrated in Fig 3). It is obvious in the bottom curve that technique with smaller δ value takes more iteration to gain best solution under noise.

The conclusion that high δ value means filter process strengthened is proven in Fig 3. Prior solutions are obtained as higher δ value.

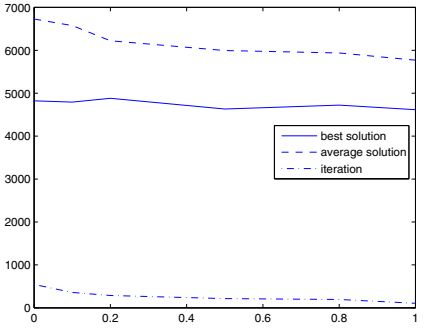


Fig. 3. Pp executions

5 Discussion and Conclusions

Decomposition is to scale down large scale CVRP into subsystems using hybrid algorithms. Cooperation and interaction is considered and solved by multi-cellular-ant algorithm in a distributed model. Potential function filters the disturbance from circumstance whose efficiency is verified from simulations. Moreover, other problem are still be improved by multi-cellular-ant algorithm, for instance, a Pork Traceability System.

Acknowledgments

This research is funded by Chongqing Natural Science Foundation (No.CSTC 2010BB2065).

References

1. Santos, L., Coutinho-Rodrigues, J., Current, J.R.: An improved heuristic for the capacitated arc routing problem. In: Computers and Operations Research, pp. 2632–2637 (2009)
2. Borgulya, I.: An algorithm for the capacitated vehicle routing problem with route balancing. Central European Journal of Operations Research, DEC, 331–343 (2008)
3. Wang, C.-H., Lu, J.-Z.: An effective evolutionary algorithm for the practical capacitated vehicle routing problems. J. Intell. Manuf. 21, 363–375 (2010)
4. Ai, T.J., Kachitvichyanukul, V.: Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. Computers and Industrial Engineering, 380–387 (2009)

5. Borgulya, I.: An algorithm for the capacitated vehicle routing problem with route balancing. Central European Journal of Operations Research 16, 331–343 (2008)
6. Yuvraj Gajpal, P.L.: Abad Multi-ant colony system(MACS) for a vehicle routing problem with backhauls. European Journal of Operational Research, 102–117 (2009)
7. Bermudez, C., Graglia, P., Stark, N., Salto, C., Alfonso, H.: Comparison of Recombination Operators in Panmictic and Cellular GAs to Solve a Vehicle Routing Problem. Inteligencia Artificial 46, 34–44 (2010)
8. Chen, Y., Liu, J.: An tree generation algorithm of undirected graphs. The application of Computer Engineer, 115–117 (2002)
9. Li, Y., Yin, Z.-m.: Techniques by compound branch and network ripping to find out all spanning trees of an undirected graph. Journal of Naval University of Engineering, 74–77 (2004)
10. Ng, A., Harada, D., Russell, S.: Policy invariance under reward transformations: theory and application to reward shaping. In: ICML 1999 (1999)
11. Xiang, Y.: Distributed structure verification in multiply sectioned Bayesian networks. In: Florida Artificial Intelligence Research Symposium, pp. 295–299 (1996)
12. Moere, A.V., Clayden, J.J.: Cellular ants: Combining ant-based clustering with cellular automata. In: International Conference on Tools with Artificial Intelligence ICTAI, pp. 177–184 (2005)
13. Yu, X.-H., Chen, G.-A., Cheng, S.-X.: Dynamic learning rate optimization of the backpropagation algorithm. In: IEEE Transactions on Neural Networks, 669–677 (1995)
14. Cao, X.-e., Chen, H.-f.: Perturbation realization, potentials and sensitivity analysis of Markov processes. IEEE Transactions of Automatic Control, 1382–1393 (1997)
15. Bagnell, J., Ng, A.: On local rewards and scaling distributed reinforcement learning. Neural Information Processing Systems (2006)
16. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. JAIR
17. Jiang, L.-y., Zhang, J., Zhong, S.-h.: (2007) Analysis of parameters in ant colony system. Computer Engineering and Applications, 31–36 (2000)
18. Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100(2), 423–445 (2004)
19. Fukasawa, R., Longo, H., Lysgaard, J., de Aragao, M.P., Reis, M., Uchoa, E., Werneck, R.F.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming 106(3), 491–511 (2006)
20. Kytojoki, J., Nuortio, T., Braysy, O., Gendreau, M.: An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Computers and Operations Research 34(9), 2743–2757 (2007)
21. Mestera, D., Braysyb, O.: Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Computers and Operations Research 34(10), 2964–2975 (2007)

Ant Colony Optimization for Global White Matter Fiber Tracking

Yuanjing Feng and Zhejin Wang

Department of Automation, Zhejiang University of Technology
Hangzhou, Zhejiang, China

Abstract. In this paper, we propose a fast and novel probabilistic fiber tracking method for Diffusion tensor imaging (DTI) data using the ant colony tracking technique, which considers both the local fiber orientation distribution and the global fiber path in collaborative manner. We first construct a global optimization model that captures both global fiber path and the uncertainties in local fiber orientation. Then, a global fiber tracking algorithm is presented using a novel learning strategy where the probability associated with a fiber is iteratively maximized. Finally, the proposed algorithm is validated and compared to alternative methods using both synthetic and real data.

Keywords: Diffusion tensor imaging, ant colony optimization, tractography, probabilistic fiber tracking.

1 Introduction

Diffusion tensor imaging (DTI) is a technique that allows measurement of white matter fiber orientation in the human brain in vivo. White matter fiber tracking or "tractography" can estimate likely fiber paths by tracing the principal diffusion directions of the local tensor orientations. These fiber paths can subsequently be used, for example, to chart the complex network of neural fiber tracts in the human brain, and to investigate connectivity in healthy and pathological populations.

One of the challenges facing conventional tractography methods is the problem of uncertainty in the tracking procedure. Noise, splitting and crossing fibers, head motion and image artifacts are all examples of factors that cause variability in the estimated fibers. As a result, stochastic tractography methods have been developed in order to quantify and visualize the uncertainty associated with the estimated fibers. Stochastic approaches model the uncertainty in DTI measurements at each voxel using a probability density function (PDF) for orientation distribution, and then trace the direction randomly sampled from the PDF [1~4]. However, these methods require the number of samples to be exponential to properly explore the state space, which is computationally burdensome.

Recently, global tractography techniques based on an iterative optimization algorithm have received considerable interest [5~8]. In the graph framework, they define a cost function to estimate the degree of a fiber trajectory belonging to the true fiber pathway running form the seed point to target point. Then optimization algorithms, such as Dijkstra algorithms [5,6,7], are used to find the maximum

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 267–276, 2011.
© Springer-Verlag Berlin Heidelberg 2011

probability fiber. These methods admit a relatively large amount of discretization error in orientation at the voxel level because they only sample the PDFs from the neighbors' directions (generally 26 directions).

In this paper, we extend a fast and novel probabilistic fiber tracking method for quantifying white matter connectivity inspired from ant colony optimization. It provides a model optimization framework for tracking the fiber directions which can capture both the local fiber orientation distribution and the global optimal maximum a posteriori fiber. In our tracking algorithm, the particles propagate the consistent orientations in a collaborative manner, which yields inherent path regularization. It provides a more accurate estimation of fiber orientations with acceptable particle numbers and reasonable computation cost.

The paper is organized as follows. In section 2, we develop the global fiber tracking optimization model between the seed point and the target region. The ant colony fiber tracking algorithm is proposed in Section 3. In Section 4, we evaluate the performance of the algorithm on synthetic data and real-world diffusion MRI data. Section 5 concludes the paper and discusses directions for future research.

2 Global Fiber Tracking

Global fiber tracking is aim to extract the most likely fiber pathway from a predefined seed point. Our formulation of extracting the most likely fiber pathway is based on computing a path of maximum probability between seed point and target region of interest. Given a fiber path $c:[x_0, x_n] \to \Omega$, where $\Omega \subset R^3$ is a compact image domain, we define the global cost function f_c of c as

$$f_c(c(x_0 \to x_n)) = \int_{x_0}^{x_n} P(T(t), c(t))\, dt \tag{1}$$

where $T(t) = c'(t) / \|c'(t)\|$ is the unit tangent vector of c. The total cost is defined as the integral of a local cost function, $P(v, x)$, and gives the cost of moving in the unit sphere direction v from the point $x \in \Omega$.

In practice, a white matter fiber path can be modeled as a trajectory of discrete points in the image space, i.e. $P_{1:n} = \{x_0, x_1, \cdots, x_n\}$. We assume that all vectors have the same step size, i.e. $\alpha_i = \alpha$, $i = 1, \cdots, n$. Hence, the growth of a path in discrete time can be described as

$$x_{i+1} = x_i + \alpha v_i, i = 0, 1, \cdots, n-1 \tag{2}$$

where x_0 is the seed point, v_i is the fiber direction at position x_i. Then the cost of (1) is approximated as

$$f(c(x_0 \to x_n)) = \alpha \sum_{i-1}^{n} P(v_i, x_i) \tag{3}$$

Here, we define the local cost function, $P(v_i, x_i)$, as the probability of selecting the direction v_i at point x_i. In deterministic tractography, such as streamline method, v_i is the largest eigenvector of the tensor D at x_i, i.e. $v_i = D_{x_i}$, and the recursion in (4) is deterministic because x_{i+1} is given by $x_i + \alpha v_i$ with probability

$$P(x_{i+1} = x_i + \alpha v_i) = 1 \tag{4}$$

Due to both noise and ambiguities for voxels where multiple fibers cross or branch, the local orientations measured by DTI are not completely reliable. Let Φ be the set of observations of a 3D diffusion weighted imaging volume. The method of probabilistic fiber tracking models this uncertainty using a probability density function (PDF) for the fiber orientations. Thus, (4) is rewritten as the stochastic recursion

$$P(x_{i+1} = x_i + \alpha v_i) = p(v_i \mid v_{i-1}, \Phi) \tag{5}$$

where $p(v_i \mid v_{i-1}, \Phi)$ is the conditional prior density under the measured diffusion tensor. We assume that the vector v_i only depends on the previous state v_{i-1}, so that

$$p(v_i \mid v_{0:i-1}, \Phi) = p(v_i \mid v_{i-1}, \Phi) \tag{6}$$

This means the new state is conditioned directly only on the immediately preceding states, and is independent of the past. Let A be a start region of interest, and B be a target region of interest. The total cost of a fiber path from A to B is

$$f(c(x_0 \in A \rightarrow x_n \in B)) = \alpha \sum_{i=0}^{n} p(v_i \mid v_{i-1}, \Phi) \tag{7}$$

However, this measure increases strongly with the path length. A uniform distribution over a range of reasonable lengths is used to describe the prior information about the expected path between A and B. In fact, the priori information about the fiber length is not easy predefined. Iturria et al. [5] define the minimum probability of $p(v_i \mid v_{i-1}, \Phi)$ as the measure. Here, we define a new measure f^a as the average weight of the fiber, i.e.

$$f^a(c(x_0 \in A \rightarrow x_n \in B)) = \frac{\alpha}{n+1} \sum_{i=0}^{n} p(v_i \mid v_{i-1}, \Phi) \tag{8}$$

We recast the problem of tracking the expected fiber path going from $x_0 \in A$ to B as that of approximating the maximum of the probability of (8),

$$f^*(c(x_0 \rightarrow B)) = \underset{\forall (x_0, \cdots, x_n \in B)}{\arg\max} \left(\frac{\alpha}{n+1} \sum_{i=0}^{n} p(v_i \mid v_{i-1}, \Phi) \right) \tag{9}$$

More formally, we define the stochastic fiber tracking problem as follow. Given a seed point $x_0 \in A$, a region B, a step length α, $P(v_i, x_i)$ at voxel x_i, and termination criteria: compute either a most likely path $c^*(x_0 \to B)$ or a set of M likely paths $C(x_0 \to B) = \{c^n(x_0 \to B)\}_{n=1}^{M}$ supported by the DT-MRI data. It can be mapped on a maximum optimization problem (S, f, Ω) that can be characterized by the following,

1) A finite set of discrete decision variables X_i, $i = 1, \cdots, n$ is defined as the voxels over the compact image domain and a set Ω of constraints include fiber curvature, FA value, et al.
2) A finite set $v_i = \left\{v_i^1, \cdots, v_i^j, \cdots, v_i^{D_i}\right\}$ of *states* of variable X_i, defined on a unit sphere.
3) A feasible solution $s \in S$, which satisfies all the constraints in the set Ω.
4) A nonempty set $s^* \in S$ of optimal solutions, with $f(s^*) \geq f(s) \quad \forall s \in S$.

where S is the set of (candidate) solutions, f is the objective function, which assigns to each candidate solution $s \in S$ an objective function (cost) value $f(s)$, and Ω is a set of constraints, which defines the set of feasible candidate solutions.

3 Ant Colony Fiber Tracking

Ant Colony Optimization (ACO) was introduced as a novel nature-inspired method for the solution of hard stochastic optimization problem. Here, we introduce ACO into global fiber tracking.

3.1 Probability Density Function of Fiber Orientation

Before going into in-depth description of our algorithm, we must discuss the fiber orientation PDFs, and select the one that we will use. In our method, we introduce an approximate observation density model by multiplying the error Gaussian distribution over all gradient directions of the diffusion weighted image, with the motivated that the noise between the observed intensity ϕ_i and the true signal s_i over direction v_i conforms closely to a normal distribution [3], i.e.

$$p(\phi_i \mid v_i) = \prod_{j=1}^{M} \frac{\mu_j}{\sqrt{2\pi\sigma^2}} e^{-\frac{\mu_j^2}{2\sigma^2}(\log\frac{s_j}{\mu_j})^2} \tag{10}$$

Here, μ_j is the observation intensity of gradient direction $j = 1, \cdots, M$, and s_j is the true signal over the gradient direction $j = 1, \cdots, M$, given by the constrained tensor model,

$$s_j = s_0 \, \mathrm{e}^{-\gamma b_j} \, e^{-\beta b_j (g_j^T v)^2} \tag{11}$$

where the gradient directions g_j and b-value b_j are the scanner parameters for data acquisition, v is the principal tensor direction, and s_0 is the intensity with no diffusion gradients applied. Parameters γ and β are determined by the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ of D with $\gamma = \frac{1}{2}(\lambda_2 + \lambda_3)$, $\beta = \lambda_1 - \gamma$.

3.2 Pheromone Representation and Diffusion

In ACO, the pheromone is represented as certain value associated with solution component, and it is stored as a table. At each iteration, when choosing a solution component, an ant uses some of the value from the table as a discrete probability distribution. In case of tractography, the choice an ant makes is the direction on a unit sphere which is not restricted to a finite set. It is impossible to represent the pheromone in the form of table. Here, we adopt a model of pheromone based on the von Mises-Fisher (vMF) distribution over a unit sphere. In fiber tracking, the particles propagate fiber paths stochastically, and the particle deposits a certain amount of pheromone trail along the fiber path. Specifically, at each step x_i, $i = 0,1,\cdots,n$, a pheromone trail g_i is assigned to fiber component along orientation v_i^j which is defined as a 3-dimentiaonal vMF function

$$g_i(x;\mu,\kappa) = \frac{\kappa^{1/2}}{(2\pi)^{3/2} I_{1/2}(\kappa)} e^{\kappa \mu^T x} \tag{12}$$

where $\kappa \geq 0, \|\mu\| = 1$, and $I_{1/2}(\cdot)$ is the modified Bessel function with order $\frac{1}{2}$. The vMF function $g_i(x;\mu,\kappa)$ is parameterized with two vectors of parameters: the means vector $v_i = v_i^j$ is the selected fiber direction, and κ is the given vector of concentration parameters.

Assume that there are m fiber paths traced by ants at each iteration. For each fiber path f_i, $i = 1,\cdots,m$, with t_i steps, we store the value of voxel coordinates and objective function $E(f_i)$. The fibers are ordered in K paths pass through voxel x_i, the pheromone diffusion process is defined as a weighted sum of these K g_i functions, which is denoted as $G_i(x)$:

$$G_i(x) = \sum_{k=1}^{K} \omega_k g_i^k = \sum_{k=1}^{K} \omega_k \frac{\kappa^{1/2}}{(2\pi)^{3/2} I_{1/2}(\kappa)} e^{\kappa \mu_i^T x} \tag{13}$$

where ω_k is the vector of weights associated with the individual Gaussian functions which is created in the following way. In our method, we keep track of a number of fibers in a fiber archive $F = [f_1, f_2, \cdots, f_n]$. For each fiber f_k with t_k steps, we store in F the value of its fiber step orientations and the value of the objective function

$E(f_k)$. Each fiber is evaluated and ranked according to $E(f_k)$, i.e., fiber f_k has rank k. The weight ω_k of the fiber f_k is calculated as

$$\omega_k = \frac{1}{\delta K\sqrt{2\pi}} e^{-\frac{(k-1)^2}{2\delta^2 K^2}} \tag{14}$$

which essentially defines the weight to be a value of the Gaussian function with argument k, mean 1.0, and standard deviation δK, where δ is a parameter of the algorithm. When δ is small, the best-ranked fibers are strongly preferred, and when it is large, the probability becomes more uniform.

3.3 Ant Colony Fiber Tracking Algorithm

In this section, we outline the ant colony fiber tracking algorithm.

Ant Fiber Propagation. Consider m particles at the starting point of a path which propagate as time progresses. The traveling of each ant in discrete time can be described by equation (2) with step size α. At step t, an ant chooses a fiber direction according to the pheromone model. As mentioned earlier, the pheromone is a mixture vMF directional model. The number of vMF functions used is equal to the size K of the fiber archive T. Sampling from a mixture vMF function is difficult. We decompose into two stages. In the first stage, we choose one of the vMF functions that compose the pheromone. The probability p_j of choosing the jth vMF function is given by:

$$p_j = \omega_j \Big/ \sum_{k=1}^{K} \omega_k \tag{15}$$

where ω_k is the weight of fiber f_k which has been defined in equation (14). Note that the choice of jth vMF function is done only once per ant, per iteration.

The second stage consists of sampling the chosen vMF function. The vMF distribution can be efficiently sampled from using the algorithm developed by Sungkyu[1] based on works of Wood [10].

Pheromone Update. The pheromone is composed of a number of vMF functions which stored as a fiber archive. This implies that the pheromone update procedure has to perform some form of update on this archive instead of pheromone evaporation.

The fiber archive T is initialized generating K fibers by sampling from (12). Then, pheromone update is conducted by adding the set of newly generated fiber to the fiber archive T and then removing the same number of worst fibers, so that the total size of the archive dose not change. This process ensures that only the best fibers are kept in the archive, so that they effectively guide the following ants toward to more likely fiber region in the search process.

To summarize, the propagating steps at each iteration are as follows.

[1] http://www.unc.edu/~sungkyu/MiscPage.html

```
1. Given step length α , seed points x0 , ants number m etc.
2. Compute the weights ωk according to (14).
3. Repeat k = 1,2,⋯
4. For i = 1:m
    while (Termination Criteria is not satisfied) do
5.    Estimate tensor model for current point.
6.     Calculate Posterior Distribution according to (10)
7.    if k = 1
8.          Draw a random direction sampling from (10)
9.     else
10.        Select the j fiber from the archive.
11.      Draw a rand direction with vMF along the fiber j .
12.        Compute the cost function according to (13)
13.      end
14.   End while
15.   Update fiber archive
16. End
```

4 Experiments and Results

In this section, we evaluate the performance of the method by experiments with both synthetic and real DTI data.

4.1 Synthetic Data

The synthetic data used in this section was created with the following parameters: $120 \times 120 \times 28$ voxels, $1 \times 1 \times 1mm$, $b = 800s/mm^2$, 6 gradient directions. We generate the observed log-signal intensities according to the constrained tensor model [3] with $\lambda_1 / \lambda_2 = 3$, $\lambda_2 = \lambda_3$. Similar to [4], we generate synthetic additive independent zero mean Gaussian noise with variance σ_r^2, where $\sigma_r = r\% \cdot (\mu_k - \mu_l)$ relates to the minimum and maximum image scalar values, μ_l and μ_k , at noise level $r\%$.

Fig. 1, panel (a)-(d) shows the results of the first experiment with synthetic two-dimensional data with complex tract configurations at noise level 25%. Panel (a) shows the data tensor field corrupted by noised. The tracking result of Friman's method with 1000 particles is shown in panel (b). Most of the fibers stop near the seed point. Panel (c) and (d) is the tracking result of our method with 50 particles after 3 and 6 iterations. Note that most of the particles can reach the target region after 6 iterations.

The second experiment is performed on fiber crossing synthetic data at noise level 25% shown in Fig.1, panel (e). As expected, the principal tensor eigenvectors in the intersection are noisy. Panel (f) shows the particle paths of a set of 1000 particles from the seed point in the left part of the horizontal fiber using Friman's Bayesian approach [3]. Panel (g) and (h) is the tracking result of our method with 50 particles after 3 and 6 iterations.

4.2 Brain Diffusion MRI

The diffusion weighted data was acquired from a real human brain using a 3-T GE system and a double echo planar imaging (EPI) sequence, with TR=17000ms, TE=78ms. The images cover a field of view of 24cm using a 144×144 grid. 85 axial slices parallel to the AC-PC line were acquired, with a slice thickness of 1.7mm. Acquisitions have 51 gradient directions with $b = 900 s / mm^2$ and 8 baseline scans with $b = 0$. The tensor parameters were estimated using weighted least squares. Spatial filtering of the tensor field was not performed.

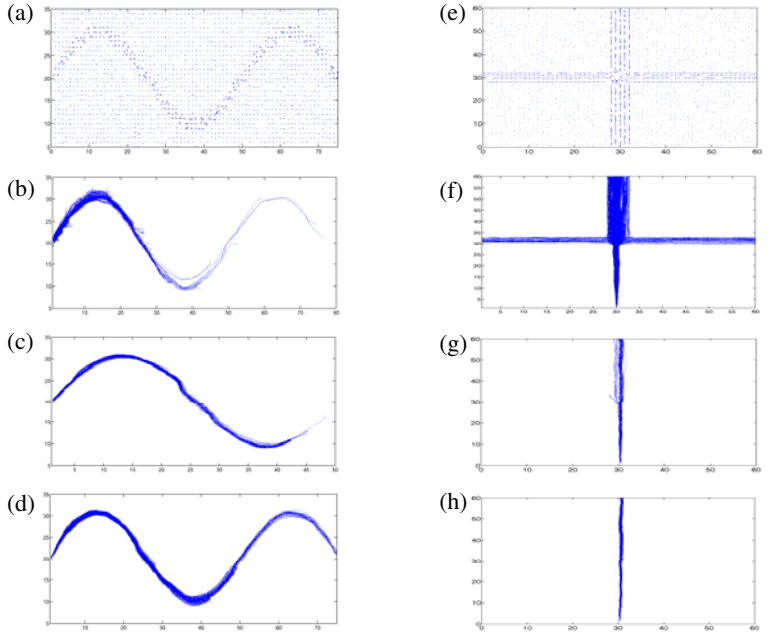


Fig. 1. (a) Synthetic data at noise level 25%. (b) Friman's method [3]. (c) our method after 3 iterations and (d) after 6 iterations. (e) Synthetic data with fiber crossing at noise level 25%. (f) Friman's method [3]. (g) Our method after 3 iterations and (h) after 6 iterations.

Fig.2 shows the experimental results with the background of a coronal slice. Fig. 2 (a), (b) and (c) show 1000 paths of Friman's method seeded separately the point in Corpus callosum and Corticopontine tract.

Fig.2 (d), (e) and (f) are the results of our method with 50 particles and 6 iterations. The figure shows that the sampled paths from Friman's method are more dispersed, with a number of paths which have low probabilities. However, our method can locate the maximum probabilities region with few particles. The result reveals how our probabilistic algorithm can address the fiber crossing and noise because of the global optimization model and the swarm collaboration.

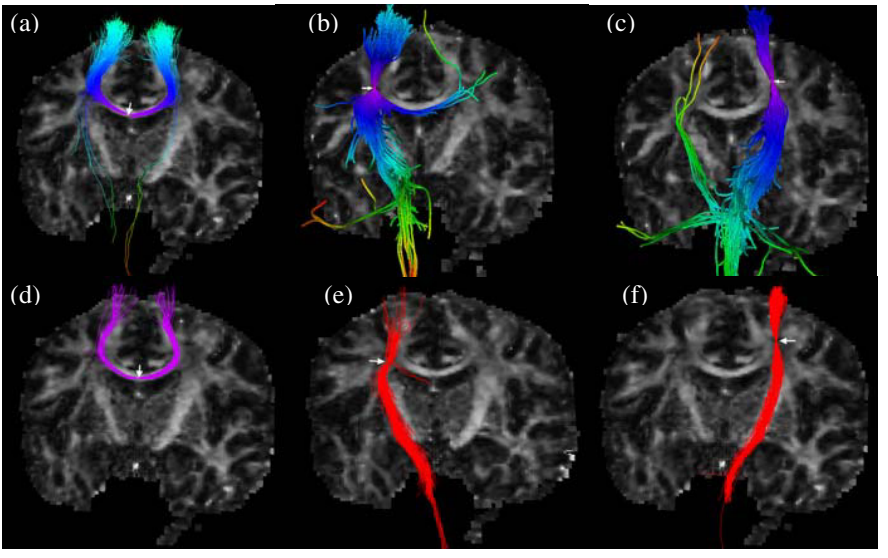


Fig. 2. (a) 1000 paths trace from a seed point in Corpus callosum using Friman's method[3](the same of (b) and (c)). (d) 50 particles 4 iterations trace from a seed point in Corpus callosum, (b) 1000 paths trace from a seed point in the right Corticopontine tract. (e) 50 particles 4 iterations trace from the same point as in (b). (c) 1000 paths trace from a seed point in the left Corticopontine tract. (f) 50 particles 6 iterations trace from the same point as in (c).

5 Conclusions

Earlier stochastic tractography algorithms use a large number of particles, many of which can deviate from the correct trajectory and terminate near the seed points. In this work, we proposed an iterative optimization approach based on a swarm tracking technique. We formulate the probability of fiber tracking as a global optimization problem. The particles are guided by both the local fiber orientation and the global direction in a collaborative manner.

From the experimental results, we can see that the advantages of the proposed algorithm are several. First, the algorithm only needs a small number of particles to rapidly locate the global optimal fiber. Second, our method can reconstruct the true fiber paths more accurately in uncertainty due to noise and fiber crossings.

The local fiber orientation distribution is based on the single tensor model which can not capture the complex fiber configurations. In future work, we will reformulate our method for the multi-tensor model of diffusion.

References

1. Basser, P., Jones, D.: Diffusion-tensor MRI: theory, experimental design and data analysis – a technical review. NMR in Biomedicine 15, 456–467 (2002)
2. Behrens, T., Woolrich, M., Jenkinson, M., Nunes, R., Clare, S., Matthews, P., Brady, J., Smith, S.: Characterization and propagation of uncertainty in diffusion-weighted MR imaging. Magnetic Resonance in Medicine 50, 1077–1088 (2003)

3. Friman, O., Farneback, G., Westin, C.: A bayesian approach for stochastic white matter tractography. IEEE Trans. Med. Imag. 25, 965–978 (2006)
4. Zhang, F., Edwin, R., Gerig, G.: Probabilistic white matter fiber tracking using particle filtering and von Mises–Fisher sampling. Medical Image Analysis 13, 5–18 (2009)
5. Iturria-Medina, Y., Canales-Rodriguez, E.J., Melie-Garcia, L., Valdes-Hernandez, E., Sanchez-Bornot, J.M.: Characterizing brain anatomical connections using diffusion weighted MRI and graph theory. NeuroImage 36, 645–660 (2007)
6. Zalesky, A.: DT-MRI Fiber Tracking: A Shortest Paths Approach. IEEE Trans. Med. Imag. 27, 1458–1471 (2008)
7. Lifshits, S., Tamir, A., Assaf, Y.: Combinatorial fiber tracking of the human brain. NeuroImage 48, 532–540 (2009)
8. Sotiropoulos, S.N., et al.: Brain tractography using Q-ball imaging and graph theory: Improved connectivities through fibre crossings via a model-based approach. Neuroimage 49, 2444–2456 (2010)
9. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Compu. 10, 440–458 (2006)
10. Wood, A.T.A.: Simulation of the von Mises–Fisher distribution. Commun. Stat. Simul. Comput. 23, 157–164 (1994)

An Efficient Bee Behavior-Based Multi-function Routing Algorithm for Network-on-Chip

Junhui Wang[1], Huaxi Gu[1], Yintang Yang[2], and Zhi Deng[1]

[1] State Key Laboratory of Integrated Service Networks
[2] Institute of Microelectronics,
Xidian University, 710071 Xi'an, China
jhwang24@gmail.com, hxgu@xidian.edu.cn, dengzhi1986@126.com

Abstract. To obtain the best food source, bees communicate their forage information by waggle dance, which indicates direction, distance, and quality of the food source they found. In this paper we propose a multi-function routing algorithm (BMFR) inspired by bees' foraging behaviors for network-on-chip (NoC). We utilize a bee agent model to exchange the states among nodes. According to these states we establish a probability model to choose the output direction. We analyze the performance of BMFR on uniform traffic pattern. Finally, we compare BMFR with XY routing algorithm on uniform and tornado traffic patterns.

Keywords: Bee agent model, Load-balance, Fault-tolerance, QoS, NoC.

1 Introduction

Swarm Intelligence has been an active area of research in the past years. In the field of communication network, Swarm Intelligence has evolved as an effective mean to solve routing problems. Many related work have already been done.

AntNet, an adaptive, distributed, mobile-agents-based multi-path routing algorithm was proposed in [1]. Encouraged by AntNet and genetic algorithm, Horst etc. proposed a multi-path routing algorithm with random decision, BeeHive [2], which is based on stochastic process and works without the necessary to save global information. Saleem etc. proposed a bee-inspired power aware routing protocol, BeeSensor [3], which requires little processing and network resources with a bee agent model. To provide guaranteed bandwidth performance for NoC, Peibo, X etc. proposed a bee-inspired QoS routing algorithm [4], in which virtual circuits and spatial division multi-plexing are employed to maintain available paths for different traffic patterns. Al Maghayreh etc. proposed a novel routing algorithm which combines Ant colony and BeeHive, called Bees-Ants [5].

Routing is a key factor that determines how much of the ideal performance could be realized. So, an appropriate routing algorithm is essential for communication networks. A large number of routing algorithms for NoC have been proposed in the past. The three types of routing algorithm are load-balance, fault-tolerance and QoS.

Arjun, S etc. proposed a load-balanced adaptive routing algorithm GOAL [6] which achieves global load balance by choosing the direction in each dimension

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 277–284, 2011.
© Springer-Verlag Berlin Heidelberg 2011

randomly. Fukushima, Y etc. proposed an effective fault-tolerance routing algorithm without virtual channels [7]. Kun, W etc. proposed a family of QoS routing algorithm with local information [8]. Each of these algorithms only considers one of the three aspects, and couldn't guarantee the performance in the others.

In this paper we propose a bee behavior-based multi-function routing algorithm, which achieves load-balance, fault-tolerance and QoS. The rest of the paper is organized as follows. Section 2 describes the bee agent model which is applied to exchange information. Section 3 describes the routing algorithm in detail. The performance of BMFR is given in section 4. In section 5, we conclude the paper.

2 The Bee Agent Model

Bee is a social insect that shows very complex behavior composed of individuals, although behavior of individual is extremely simple. Generally speaking, the model of bee colony intelligence consists of three basic elements: food source, employed foragers and unemployed foragers. The two of most basic models of behavior are recruiting bee for certain food source and abandoning the certain food source. To choose the best food source, bees exchange their information about food source by dancing in cellular.

The organizational principles of honey bee are helpful to solve routing problems in NoC. We borrow the concept from bees' communication principles to design the bee agent model, which exchanges information between neighbor nodes. The information is used to estimate the probability which determines the routing direction.

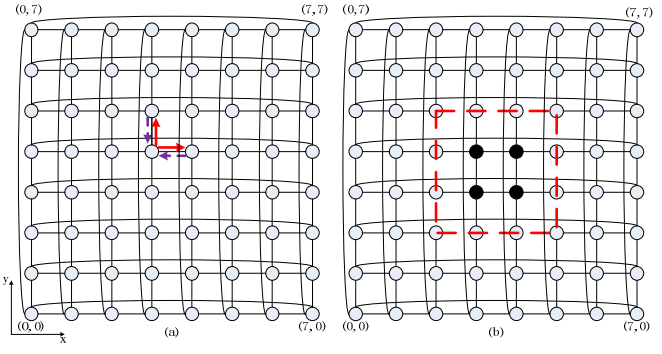


Fig. 1. (a) The 8×8 torus topology, the edge nodes transmit bee agents through these long links. (b) The fault region we assumed (*the dashed line is the fault ring*).

There are three types of bee agents in this model: forward-bee, backward-bee and reference-bee. The forward-bee and backward-bee work as follows.

Firstly, two forward-bees (play the role of forager, the solid line in Fig.1.(a)), with the number of free buffer at each port (just like the information of food source), are launched by each node to the neighbor nodes in the positive direction (x+ and y+) through the control network periodicity. Afterwards, the neighbor nodes read the information carried by the forward-bees (as bees dance in cellular). Then, the

neighbor nodes rewrite their own free buffer number to the forward-bees to construct the backward-bees (the dashed line in Fig.1.(a)), and send them back to the previous nodes in the negative direction (x- and y-) through the control network. Finally, the previous nodes get the information from backward-bees. Then, the backward-bee is killed. Thus each node obtains free buffer number of the four neighbor nodes.

The reference-bee is applied to transmit the coordinate of reference nodes to these nodes on the fault ring, which is defined in 3.2. After dancing in cellular, all of the reference-bees are killed.

With the three bee agents, every node in the network could get the information of its neighbor nodes or some special nodes through the control network.

3 Multi-function Routing Algorithm

A detailed description about the multi-function routing algorithm we proposed is given in this section. This routing algorithm combines bee agent model, improved GOAL and fault-tolerance routing algorithm. Besides, we also apply different types of packets to provide better QoS.

3.1 Load Balance and QoS

A good routing algorithm should provide low latency, high throughput and balanced load on adversarial traffic patterns. In BMFR, the concept quadrant inspired by GOAL is applied to provide global load balance and the probability based minimal routing is applied to provide local load balance in the selected quadrant. Besides, we add priorities for different types of packets to provide better QoS.

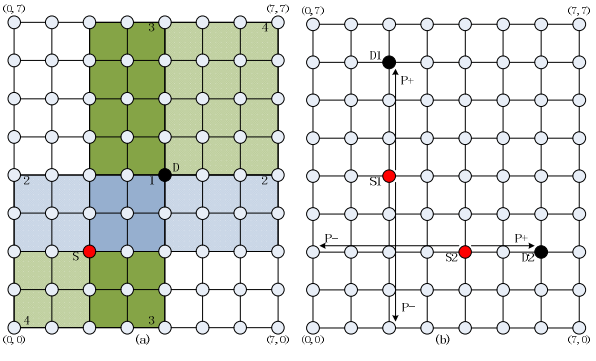


Fig. 2. (a) Source and destination nodes in different row and column. (b) Source and destination nodes in the same row or column. For simplicity, we use mesh instead of torus.

Based on the distance between source and destination nodes, a direction is chosen in each dimension. Thus, the quadrant in which a packet will be transmitted was determined after the packet was generated in the source node. How to determine the direction and quadrant are described as follows.

a): As is shown in Fig.2.(a), source and destination nodes in different row and column, equation (1)-(4) are applied to determine the quadrant.

b): As is shown in Fig.2.(b), source and destination nodes in the same row or column. In the same row, equation (2), else equation (3) is applied to determine the direction.

$$offset = D - S = (\Delta x, \Delta y) \ . \tag{1}$$

$$P_{x-} = \frac{\min(|\Delta x|, k_x - |\Delta x|)}{k_x} \ , \quad P_{x+} = \frac{\max(|\Delta x|, k_x - |\Delta x|)}{k_x} \ . \tag{2}$$

$$P_{y-} = \frac{\min(|\Delta y|, k_y - |\Delta y|)}{k_y} \ , \quad P_{y+} = \frac{\max(|\Delta y|, k_y - |\Delta y|)}{k_y} \ . \tag{3}$$

$$P_{q1} = P_{x+} \times P_{y+} \ , \ P_{q2} = P_{x-} \times P_{y+} \ , \ P_{q3} = P_{x+} \times P_{y-} \ , \ P_{q4} = P_{x-} \times P_{y-} \ . \tag{4}$$

Where D stands for the coordinate of the source node, S for the destination node. P_{x-} stands for the probability that choose the longer path in x-coordinate, P_{x+} for the shorter one. P_{qi} stands for the probability that choose quadrant i.

In order to provide lower latency for special information, we assume that all the packets are divided into four categories. The packets with most stringent requirement of latency, such as video traffic, have the highest priority. For these packets, quadrant 1 is always the choice. The others follow the probabilities which get from equation (4) to determine the quadrant. Thus, we can provide better global load balance and lower latency for these packets need to be transmitted as soon as possible.

$$f_x + f_y \neq 0 \quad P_x = \frac{f_x}{f_x + f_y} \ , \quad P_y = \frac{f_y}{f_x + f_y} \ . \tag{5}$$

$$f_x + f_y = 0 \quad P_x = 0 \ , \quad P_y = 0 \ . \tag{6}$$

P_x stands for the probability that choose direction in x-coordinate, P_x for y-coordinate.

In the selected quadrant, the probability based minimal routing algorithm is applied to provide local load balance and lower latency. For each node, there are two available directions (x or y) to transmit the packet in the shortest path. We choose one according to the probabilities which estimated through free buffer number transmitted by the bee agents and the priority of the packet. The probabilities are estimated by equation (5) and (6), where f stands for free buffer number of the corresponding input port of the neighbor node. We define $f = 0$, in the case that neighbor node is a faulty one. When competition emerges, the direction with higher probability is chosen for the packet with higher priority. Thus, because of the decreasing of the latency for these packets with high priority, we can get better performance.

3.2 Fault Tolerance

As device shrinks toward the nanometer scale, on-chip circuits are becoming vulnerable to errors. So, a fault-tolerance routing algorithm is essential to provide reliable communication on unreliable physical interconnects. In BMFR, we propose a protocol to transmit packets bypass the fault region with minimal hops. To reduce the number of disabled nodes, we adopt the node deactivation algorithm proposed in [7] to form the fault region. For torus, all of the nodes are equal in the network, the fault region is shown in Fig.1.(b).

In order to bypass the fault ring with minimal hops, we define the northeast corner node of the faulty ring as N-reference node, the southwest one as S-reference node. The reference-bee is applied to transmit the coordinate of the reference nodes to these nodes on the fault ring. It's clear that a packet encounters the fault region only when the current and the destination nodes in the same row or column. The fault-tolerance routing algorithm is described as follow.

a): Current node on the left (right) of the fault ring.
Clockwise (counter-clockwise) is the choice when N-reference node is closer than S-reference node in y-coordinate, otherwise counter-clockwise (clockwise).

b): Current node on the top (bottom) of the fault ring.
Clockwise (counter-clockwise) is the choice when N-reference node is closer than S-reference node in x-coordinate, otherwise counter-clockwise (clockwise).

In BMFR, deadlock could not be avoided. To solve this problem, a timer is applied. The timer starts to work when a packet arrives at the router. In the case that the packet transmitted unsuccessfully before timeout, we assume deadlock emergence. Then the packet will be sent to the local IP core. After a random time, the packet will be retransmitted by the router.

3.3 Pseudo Code

The following pseudo code describes the proposed routing algorithm in detail.

```
/*S is source, D is destination, C is current node */
/*choose quadrant or direction*/
if(priority is 1)
 if(Source and destination in different row and column)
      choose quadrant 1;
   else
      choose direction with higher probability;
else
   choose quadrant/direction according to probabilities;
/*probability-based minimal routing*/
if(C is destination D)
   send P to local IP core;
else if(C and D in the same row or column)
   if(C on the fault ring)
      fault-tolerance routing;
   else
      normal routing;
```

```
else
   normal routing;
/*normal routing*/
if(priority is 1)
   send P to the direction with higher probability;
else
   send P according to probabilities;
/*fault-tolerance routing*/
if(C on the left (right) of the ring)
   if(N-reference closer in y-coordinate)
      clockwise(counter-clockwise);
   else
      counter-clockwise(clockwise);
else if(C on the top (bottom) of the ring)
   if(N-reference closer in x-coordinate)
      clockwise(counter-clockwise);
   else
      counter-clockwise(clockwise);
```

4 Performance Evaluation

In order to analyze the performance of BMFR, we simulate the results on the 8×8 torus topology. Each node operates asynchronously and generates packets at time interval chosen from a negative exponential distribution. The simulation is completed with the network simulator-OPNET. Wormhole switching is chosen as the switching mechanism. We analyze the performance of the network with 6% faulty nodes in terms of latency and throughput on uniform traffic pattern. We also compare BMFR with XY routing algorithm on uniform and tornado traffic patterns in the case that all nodes are working well.

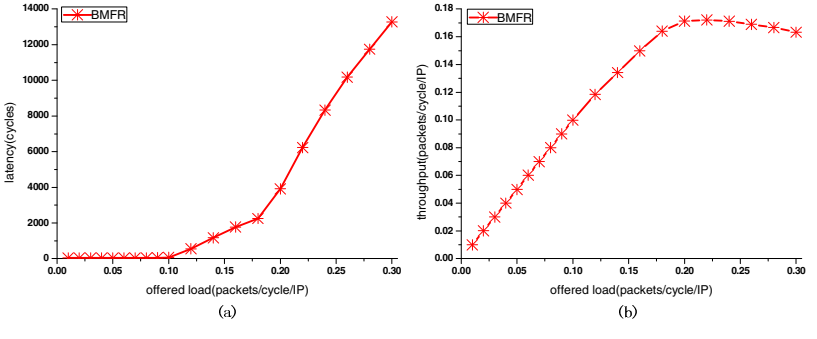


Fig. 3. (a) Latency of BMFR on uniform traffic. (b) Throughput of BMFR on uniform traffic.

The performance of BMFR with 6% faulty nodes on uniform traffic pattern is shown in Fig.3. It shows that BMFR has good performance when offered load is low. In Fig.3.(a), we can see that the latency keeps small and stable until offered load reaches 0.10. Then, the latency rises sharply with the growth of offered load. In Fig.3.(b),

we can see that the network reaches saturation point when the offered load is 0.22. But the latency is really a large one at the saturation point.

Fig.4.(a) and (b) show the performance of BMFR and XY routing algorithm on uniform traffic, (c) and (d) on tornado traffic pattern without faulty nodes. A good routing algorithm should provide stability on adversarial traffic patterns. It's clear that BMFR is more stable than XY routing algorithm on different traffic patterns. It means that BMFR has adaptability benefit of XY routing algorithm.

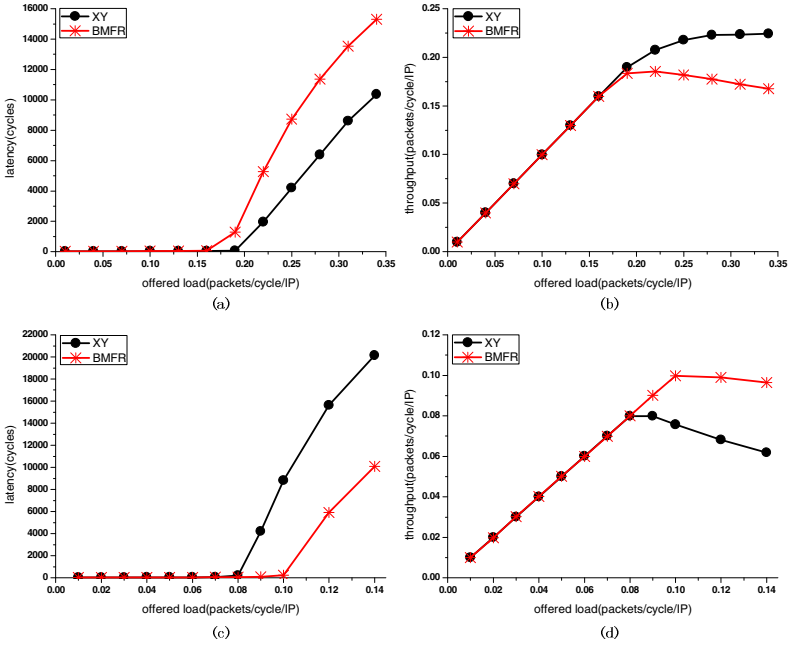


Fig. 4. (a) Latency of BMFR and XY routing algorithm on uniform traffic. (b) Throughput of BMFR and XY routing algorithm on uniform traffic. (c) Latency of BMFR and XY routing algorithm on tornado traffic. (d) Throughput of BMFR and XY routing algorithm on tornado traffic.

5 Conclusions

In this paper we proposed BMFR for NoC. Bee agent model is applied to exchange information among nodes. Based on quadrant and probability based minimal routing we get better load balance. The priority of packets is applied to achieve lower latency for these packets with high requirement on latency. The fault-tolerance routing algorithm provides reliable communication on unreliable physical interconnects. For general purpose, BMFR contains all of the three main aspects of a routing algorithm. We analyzed the performance of BMFR with OPNET simulator. Simulation results show the latency and throughput under different offered loads and distribution probabilities. In the future, we will consider the case that more than one fault ring emerge and online fault detection.

Acknowledgments. This work is supported partly by the National Science Foundation of China under Grant No.60803038, No.61070046 and 60725415, the special fund from State Key Lab (No.ISN090306), the Fundamental Research Funds for the Central Universities under Grant No.K50510010010 and the 111 Project under Grant No. B08038.

References

1. Caro, G.D., Dorigo, M.: AntNet: Distributed stigmergetic control for communication networks. J. Artificial Intelligence Research 9, 317–365 (1998)
2. Wedde, H.F., Farooq, M., Zhang, Y.: BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 83–94. Springer, Heidelberg (2004)
3. Saleem, M., Farooq, M.: BeeSensor: A Bee-Inspired Power Aware Routing Protocol for Wireless Sensor Networks. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 81–90. Springer, Heidelberg (2007)
4. Peibo, X., Huaxi, G.: Intelligent bees for QoS routing in Networks-on-Chip. In: 2nd Pacific-Asia Conference on Circuits, Communications and System (PACCS), pp. 311–314. IEEE Press, Beijing (2010)
5. Al Maghayreh, E., Al-Haija, S.A., et al.: Bees_Ants Based Routing Algorithm. In: International Conference on Intelligent Systems, Modelling and Simulation (ISMS), pp. 344–349. IEEE Press, Liverpool (2010)
6. Arjun, S., Dally, W.J., et al.: GOAL: a load-balanced adaptive routing algorithm for torus networks. In: 30th Annual International Symposium on Computer Architecture, pp. 194–205 (2003)
7. Fukushima, Y., Fukushi, M., et al.: Fault-Tolerant Routing Algorithm for Network on Chip without Virtual Channels. In: 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 313–321. IEEE Press, Chicago (2009)
8. Kun, W., Changshan, W., et al.: Quality of service routing algorithm in the torus-based network on chip. In: 8th IEEE International Conference on ASIC, pp. 952–954. IEEE Press, Changsha (2009)

Artificial Bee Colony Based Mapping for Application Specific Network-on-Chip Design

Zhi Deng[1], Huaxi Gu[1], Haizhou Feng[2], and Baojian Shu[2]

[1] State Key Laboratory of Integrated Service Networks,
Xidian University, 710071 Xi'an, China
[2] ZTE Corporation, Shenzhen, China
dengzhi1986@126.com, hxgu@xidian.edu.cn

Abstract. A new mapping algorithm is proposed based on Artificial Bee Colony (ABC) model to solve the problem of energy aware mapping optimization in Network-on-Chip (NoC) design. The optimal mapping result can be achieved by transmission of the information among various individuals. The comparison of the proposed algorithm with Genetic Algorithm (GA) and Max-Min Ant System (MMAS) based mapping algorithm shows that the new algorithm has lower energy consumption and faster convergence rate. Simulations are carried out and the results show the ABC based method could save energy by 15.5% in MMS, 5.1% in MPEG-4 decoder and 12.9% in VOPD compared to MMAS, respectively.

Keywords: Network-on-Chip(NoC), mapping optimization, energy consumption.

1 Introduction

With the development of semiconductor technology, the number of Intellectual Property (IP) cores integrated on a single chip is increasing dramatically. It is possible for a single chip to integrate a system. Finally, System-on-Chip (SoC) appears.

However, with the further progress in system integration, the existing shared bus structure faces enormous challenges in performance, such as delay, throughput, power consumption, synchronization, scalability and so on. Fortunately, an emerging research field, Network-on-Chip (NoC), endeavors to solve these issues above. As soon as it is proposed, NoC has been attracted more attentions for some new mechanisms introduced. For example, it uses wormhole switching as basic switching mechanism, and Global Asynchronous Local Synchronous (GALS) as clock mechanism which can solve the single global clock synchronization challenge.

Application specific mapping from IP core to architecture is a key step in NoC design, which significantly affects the system performance, such as energy, latency and load balance. However, the mapping optimization is a NP-complete problem and difficult to obtain optimal solution by traditional methods due to the complexity of time and space. Therefore, many scholars settle this problem by using heuristic algorithm, such as Branch and Bound(BB)[1], Ant Colony Algorithm(ACA) [2], Genetic Algorithm(GA)[3], and Particle Swarm Optimization(PSO)[4] algorithm

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 285–292, 2011.
© Springer-Verlag Berlin Heidelberg 2011

et al. However, those methods show low convergence speed and the optimal solution is always affected by local solution. On the contrary, Artificial Bee Colony(ABC) algorithm performs lower complexity and better performance, for example, it not only converges with a rapid speed, but is hardly affected by local optimal solution [5][6].

In this paper, we address energy aware mapping from IP core to NoC platform. First of all, the energy model associated to 2D mesh topology and mapping problem is formulated. Then, an efficient colony algorithm is proposed to map IP cores to nodes of NoC, such that the total energy consumption is minimized. This method shows a better performance and also can be applied to the other topologies.

2 Related Work

Hu et al. [1] proposed a Communication Weighted Model (CWM) by constructing the graph of IP core as vertex and communication volume as weight. They adopted Branch-and-Bound (BB) algorithm to solve optimization model in the target of energy such that bandwidth constraint. According to simulation results, the proposed method had advantages over random mapping and decreased 60% of communication energy. Zhou et al. [2] used improved ant colony algorithm to solve optimization model with objective of power consumption. Due to the development in the convergence of Max-Min Ant System (MMAS) [7] algorithm, this method could achieve better performance. Experimental results showed that 25% to 70% of power consumption was reduced depending on different applications. Tang et al. [3] suggested a two-step Generic Algorithm (GA) based approach to map from the parameterized task graph to NoC architecture with 2D mesh topology. They believed that the method could be able to handle large task graph and provide near optimal mapping in a few minutes. Wang et al. [4] proposed a heuristic two step strategy to map tasks to tiles on NoC platform in the target of energy consumption and delay. They tried to assign the tasks to IP cores firstly, and then mapped from IP core to tile on architecture by chaotic Discrete Particle Swarm Optimization (DPSO) method. They said that the solution obtained by DPSO was 6.852% better than what was obtained by GA.

3 Problem Formulation

Definition 1. Given a communication core graph(CCG), where vertex $t_i \in T$ represents a IP core in the application specific communication, a directed arc $c_{i,j} \in C$ represent communication between the IP core t_i and t_j, and the weight of edge $v_{i,j} \in V$ represents communication volumes from the IP core t_i to t_j.

Definition 2. Given a topology architecture graph (TAG) $G(N, P, E)$, where vertex $n_i \in N$ represents a tile in the architecture, a directed edge $p_{i,j} \in P$ represents routing path from the tile n_i to tile n_j, and $e_{i,j} \in E$ in the architecture represents average energy consumption of sending one bit of data from the source tile n_i to destination tile n_j.

3.1 Energy Model

We use the model for energy consumption proposed in [8]. The energy (E_{bit}) consumed when one bit of data is transported from a tile to its neighbor is defined as

$$E_{\text{bit}} = E_{S_{\text{bit}}} + E_{B_{bit}} + E_{W_{bit}} + E_{L_{bit}}. \tag{1}$$

where $E_{S_{bit}}$, $E_{B_{bit}}$, $E_{W_{bit}}$ and $E_{L_{bit}}$ represent the energy consumed by the switch, buffering, interconnection wires inside switching fabric and links, respectively.

Since the length of a link is typically in the order of millimeters on chip, the energy consumed by buffering($E_{B_{bit}}$) and internal wires($E_{W_{bit}}$) is negligible compared to $E_{L_{bit}}$ [9]. So equation (1) can be defined approximately as

$$E_{\text{bit}} = E_{S_{\text{bit}}} + E_{L_{bit}}. \tag{2}$$

Then average energy consumption per bit data from tile n_i to n_j is

$$E_{bit}^{n_i,n_j} = H \times E_{S_{bit}} + (H+1) \times E_{L_{bit}}. \tag{3}$$

where H represents hops of bit traversing from tile n_i to n_j . Generally speaking, the H in regular topology could be Manhattan distance instead.

3.2 Optimization Model

With the model mentioned in 3.1, the optimal mapping problem of minimizing communication energy consumption can be described as

$$\min\{Energy = \sum\nolimits_{\forall c_{i,j}} v_{i,j} \times E_{bit}^{map(t_i),map(t_j)}\}. \tag{4}$$

s.t.

$$\forall t_i \in T, map(t_i) \in N. \tag{5}$$

$$\forall t_i \neq t_j \in T, map(t_i) \neq map(t_j) \in N. \tag{6}$$

where $map(t_i)$ represents the mapping result of IP core t_i and $E_{bit}^{map(t_i),map(t_j)}$ is the average energy consumption per bit. Conditions (5) and (6) mean that each IP core should be mapped to one tile and no tile can host more than one IP core.

4 ABC Based Mapping

As is known to all, a colony of honey bee is composed of a queen, some drones and thousands of workers. The mission of the queen is to lay eggs as many as possible and make new colonies. The drones' most important work is to mate with queen in order to help queen produce more offspring. The smallest but most individual in a colony is

worker bee. The worker bees are responsible for building honeycomb, taking caring of the young, feeding the queen and drones, collecting foods and so on.

The ABC algorithm derives from the process of worker bees searching food for the individuals' cooperation. It was firstly presented by Seely in 1995 and then improved by D. Karaboga in 2005 [10]. Generally speaking, three kinds of worker bees, leader bees, scout bees and follower bees, are in charge of the food and information collection. In other words, the optimal solution can be achieved by the following behavior in search space via the Artificial Bee Colony (ABC) algorithm [6].

4.1 The Behavior of Leader Bee

The responsibility of leader bees is leading the followers to certain food source. So, one food source has a leader bee at least and all the leaders should know the situation of their food. Also, the leader bees should tell the followers necessary information about quality and scale of the food they known. According to the experience, the proportion of leader bees in the colony is 50% in this method we used.

4.2 The Behavior of Scout Bee

The Scouts fly around and search for food randomly. Once find, the scout bee will fly back to report the information of the food by dancing as soon as possible. In ABC algorithm, the scout bee must finish the activities including searching solution randomly and reporting necessary information. In this paper, we choose 10% individuals of the colony as the scout bees.

4.3 The Behavior of Follower Bee

The Followers must decide which food source to go to when they get the information from the leader bee. Most individuals do select the best food source finally and few of them do not for some uncertain factors. In order to simulate the behavior of follower bee, the artificial follower bees must be responsible for the selection of solution according to the current information, and few of them also need to go to the non-best location with roulette simultaneously. The ratio of follower bees is 40%.

4.4 The Pseudo Code

The following pseudo code is to describe the proposed algorithm in detail:

```
ABC based mapping
  const  colony_size = 100;// size of colony
          max_iter = 1000;// max iterations
          limit = 50;// max number without changed
          mut_prob = 0.1;// mutation probability
  generate initial colony;
  begin
    iter := 1;
    repeat
      sort all individuals in ascending order of cost;
      //cost relates to energy consumption
      leader bee behavior();
```

```
    scout bee behavior();
    follower bee behavior();
    if number of individual k without changed  > limit
      individual k := generate new individual randomly;
    endif.
    for individual i
     //mutate all colony with a probability of 0.1
      if random number > mut_prob
        individual i:= generate new individual randomly;
      endif.
    endfor.
    record best individual in all colony;
    iter := iter + 1;
  until iter = max_iter
end.
```

5 Experiments and Results

5.1 Experimental Descriptions

In order to show the performance of the proposed mapping algorithm, we use Genetic algorithm(GA) [3], MAX-MIN Ant System(MMAS) [2] and Artificial Bee Colony (ABC) based mapping to different benchmarks, respectively. The simulations are done with MATLAB in windows XP OS on a computer with Pentium(R) Dual-Core CPU E6500 @2.93GHz 2.93GHz, 1.96GB RAM. In this paper, the Bit energy consumption for link and switch are 0.7066(nJ) and 0.9334(nJ) at technology of 1GHz and 0.18µm, respectively [11].

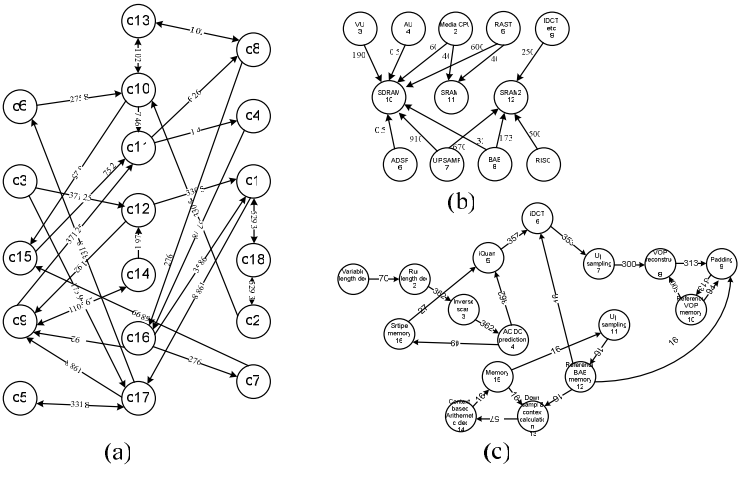


Fig. 1. The benchmark of application specific traffic: (a) MMS communication core graph (Mb/s); (b) MPEG-4 decoder communication core graph (Mb/s); (c) VOPD communication core graph (Mb/s)

As is shown in Fig. 1, they are three application specific communication core graphs used in most of simulations in application specific NoC design[1][2][8][9][12]. Fig. 1(a) is a Multi Media System (MMS) communication graph and the system tasks are partitioned into 40 tasks that are assigned onto the 18 IP cores [13]. In this paper, we will map from MMS with 18 cores onto 3×6 2D mesh topology architecture. The MPEG-4 decoder has been used for testing purposes in the past [14] and the application core graph with 12 IPs is shown in Fig. 1(b) [15]. It will be mapped onto 3×4 2D mesh topology. The Video Objective Plane Decoder (VOPD) core graph with 16 IPs [15] will be mapped onto 4×4 2D mesh topology is shown in Fig. 1(c).

5.2 Experimental Results

Different optimization algorithms lead to different performances and runtimes in the process of getting optimal solution. Fig. 2 shows the optimal solution converged by GA, MMAS and ABC based mapping during iteration process in different benchmarks respectively. It is easy to conclude that the ABC based mapping shows a

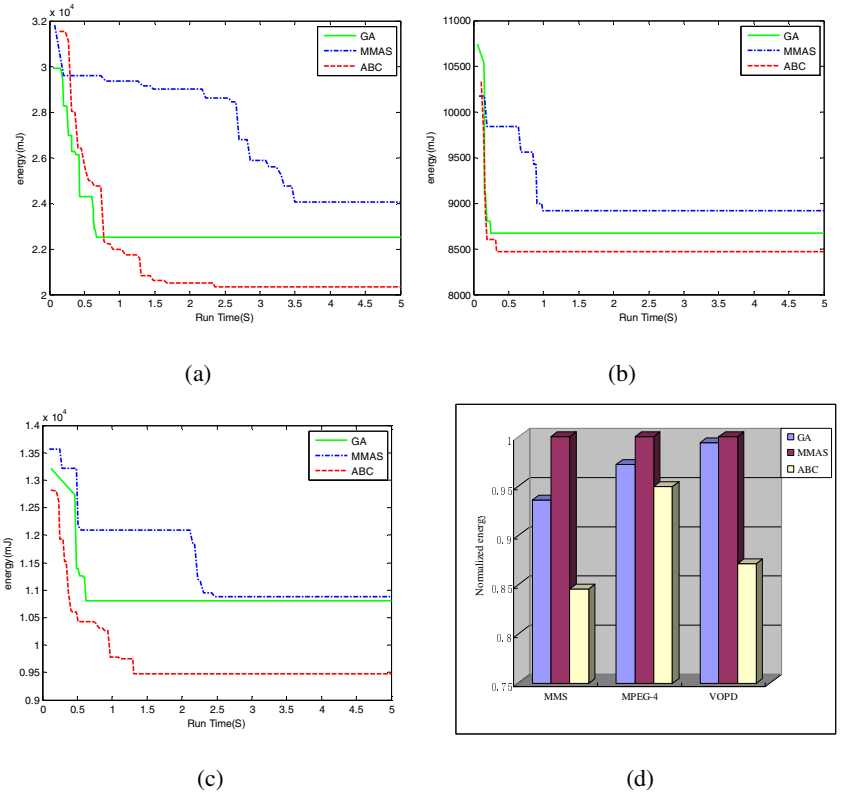


Fig. 2. The performance of different methods during iteration process: (a) the energy consumption in MMS; (b) the energy consumption in MPEG-4 decoder; (c) the energy consumption in VOPD; (d) the normalized energy consumption of the final results.

faster convergence speed than GA and MMAS. Perhaps, GA based mapping could have got a global optimal solution if it could maintain the convergence rate. Unfortunately, it always falls into the local optimization and does not get a better solution. The convergence of MMAS based mapping is so slow that it needs much time to run in order to get better mapping performance. Moreover, it is sensitive to the local optimal solution.

Fig. 2(a) shows a significant reduction of energy consumption with runtime by using the ABC based mapping compared to GA and MMAS. The mapping result of ABC is 20329.28mJ in MMS, but the GA is 22500.49mJ and MMAS is 24049.83mJ. The results of three mapping methods in MPEG-4 decoder are shown in Fig. 2(b). Precisely, the energy consumption of ABC, MMAS and GA based mapping methods are 8471.16mJ, 8919.26mJ and 8669.6mJ respectively. In VOPD, it is conclude that the energy consumption of the ABC based mapping saves more than GA and MMAS based mapping. As is shown in Fig. 2(c), the energy consumption is 9470.21mJ with the ABC, 1087.4mJ with MMAS and 10803.52mJ with GA based mapping finally. Fig. 2(d) shows the normalized energy consumption of the final results with different mapping methods, the energy consumption of MMAS method is unitary in this figure. According to the calculation analysis, the energy consumption of AFSA saves 15.5% in MMS, 5.1% in MPEG-4 decoder and 12.9% in VOPD compared with MMAS based mapping.

6 Conclusions and Future Works

In this paper, an artificial bee colony (ABC) based mapping method is presented in application specific NoC design. According to the simulation results of different benchmark applications, the proposed mapping method shows a better performance than GA and MMAS, such as lower energy consumption and faster convergence rate. The energy of ABC based mapping can save 15.5% in MMS, 5.1% in MPEG-4 decoder and 12.9% in VOPD compared to MMAS, respectively. In the future, more work needs to be done in order to improve global search capability.

Acknowledgments. This work is supported partly by the National Science Foundation of China under Grant No.60803038, No.61070046 and 60725415, the special fund from State Key Lab (No.ISN090306), the Fundamental Research Funds for the Central Universities under Grant No.K50510010010, the 111 Project under Grant No. B08038 and ZTE University cooperation project.

References

1. Jingcao, H., Radu, M.: Energy-aware mapping for tile-based NoC architectures under performance constraints. In: The 2003 Asia and South Pacific Design Automation Conference, pp. 233–239. ACM, Kitakyushu (2003)
2. Zhou, G., Yin, Y., Hu, Y., Gao, M.: NoC Mapping Based on Ant Colony Optimization Algorithm. Computer Engineering and Applications 41(18), 7–10 (2005) (in Chinese)

3. Lei, T., Kumar, S.: A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In: Euromicro Symposium on Digital System Design, pp. 180–187 (2003)
4. Lei, W., Xiang, L.: Energy- and Latency-Aware NoC Mapping Based on Chaos Discrete Particle Swarm Optimization. In: 2010 International Conference on Communications and Mobile Computing (CMC), pp. 263–268 (2010)
5. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. Applied Soft Computing 8(1), 687–697 (2008)
6. Ding, H., Li, F.: Bee Colony Algorithm for TSP Problem and Parameter Improvement. China Science and Technology Information 03, 241–243 (2008) (in Chinese)
7. Stüzle, T., Hoss, H.H.: MAX-MIN Ant system. Future Gener. Comput. System. 16(9), 889–914 (2000)
8. Hu, J., Marculescu, R.: Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures. In: Design, Automation and Test in Europe Conference and Exhibition, pp. 688–693 (2003)
9. Hu, J., Marculescu, R.: Energy- and performance-aware mapping for regular NoC architectures. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24(4), 551–562 (2005)
10. XingBao, L., ZiXing, C.: Artificial Bee Colony Programming Made Faster. In: Fifth International Conference on Natural Computation (ICNC 2009), pp. 154–158 (2009)
11. Chen, X., Peh, L.-S.: Leakage power modeling and optimization in interconnection networks. In: The 2003 International Symposium on Low Power Electronics and Design (ISLPED 2003), pp. 90–95 (2003)
12. Van Der Tol, E.B., Jaspers, E.G.T.: Mapping of MPEG-4 decoding on a flexible architecture platform. In: SPIE - Medio. Processors, pp. 1–13 (2002)
13. Morgan, A.A., Elmiligi, H., El-KharashiF, M.W.,Gebali, F.: Multi-objective optimization for Networks-on-Chip architectures using Genetic Algorithms. In: 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 3725–3728 (2010)
14. Murali, S., De Micheli, G.: SUNMAP: a tool for automatic topology selection and generation for NoCs. In: 41st Proceedings of Design Automation Conference, pp. 914–919 (2004)
15. Dumitriu, V., Khan, G.N.: Throughput-Oriented NoC Topology Generation and Analysis for High Performance SoCs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 17(10), 1433–1446 (2009)

Using Artificial Bee Colony to Solve Stochastic Resource Constrained Project Scheduling Problem

Amin Tahooneh[1] and Koorush Ziarati[2]

[1] Department of Mathematics, College of Science, Science,
Shiraz University, Shiraz, Iran
[2] Department of Computer Science and Engineering,
Shiraz University, Shiraz, Iran
amin.tahooneh@gmail.com, ziarati@shirazu.ac.ir

Abstract. Resource constrained project scheduling (RCPSP) is one of the most crucial problems in project problem. The aim of RCPSP, which is NP-hard, is to minimize the project duration. Sometimes the activity durations are not known in advance and are random variables. These problems are called stochastic resource constrained project scheduling problems or stochastic RCPSP. Various algorithms such as genetic algorithm and GRASP have been applied on stochastic RCPSP. Bee algorithm is a metaheuristic based on the intelligent behavior of honey bee swarms. The goal of this study is adopting the artificial bee colony (ABC) algorithm to solve stochastic RCPSP and investigating its performance on the stochastic RCPSP. Simulation results show that proposed algorithm is an effective method for solving the stochastic resource constrained project scheduling problem. With regard to the problems with high distribution variability, the ABC algorithm is more effective than the other algorithms in the literature.

Keywords: Artificial Bee Colony, stochastic RCPSP, Serial-SGS, Stochastic Serial-SGS.

1 Introduction

Resource constrained project scheduling problem is one of the major problems in project management. It has been shown that this problem is the NP-hard [4]. During the project execution, each activity requires many resources. Since the capacity of each resource is limited, the proper use of resources will reduce the makespan of the project. These projects, which focus on the minimization of project duration subject to resource and precedence constraints, have $n + 2$ activities and m renewable resources. In all scheduling problems, activity durations are not deterministic because during the project implementation there may be a series of random factors affecting the duration of activities. These uncertainties are derived from several factors such as an increase or decrease in the originally estimated time, unavailability of resources, late material arrivals, the absence of workers, network-structure changes and the bad weather delays. In this article, a case in which durations are not known beforehand has been considered. The durations are random variables. The objective of stochastic RCPSP is

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 293–302, 2011.
© Springer-Verlag Berlin Heidelberg 2011

to minimize the expected makespan. Now there exists an important question: What is the feasible solution to the stochastic RCPSP when a deterministic problem becomes a stochastic problem. Necessarily a deterministic scheduling does not provide us with sufficient information about creating a feasible solution. Unlike the events that occur during the project, the represented solution should express the appropriate behavior. Such solutions are called "Policy" [9]. Several classes of the policies were examined by Stork [14]; the classes of the so-called activity-based policies were utilized for large instances. Also, this policy is used in this paper. Here, the stochastic serial-SGS is applied in order to find a feasible solution to the stochastic RCPSP [2].

There are a few heuristic algorithms for the stochastic RCPSP. Research into stochastic RCPSP, however, has remained limited to date, with few computational publications addressing this problem: Igelmund and Radermacher [9] and stork [14] reported on experiments using branch-and-bound algorithms, while Golenko-Ginzburg and Gonik [6] and Tsai and Gemmill [15] developed greedy and local search heuristics. Time/resource trade-offs with stochastic activity durations, in which the resource allocation influences the mean and/or the variance of the durations, have been investigated by Gerchak [5] Gutjahr [7] and Wolmer [17]. This work aims at presenting a heuristic algorithm for the stochastic RCPSP. In this paper, the artificial bee colony (ABC) algorithm is used and a method for finding a solution to a stochastic RCPSP is investigated. To minimize the expected makespan of a stochastic RCPSP, meta-heuristics inspired from the collective behavior of honey bees.

The remainder of this paper is organized as follows: in Section 2, the definition of the problem and the formulation of the stochastic RCPSP are presented; in section 3, the artificial bee colony algorithm is explained. Next, Section 4 explains the application of ABC algorithm for stochastic RCPSP. Then, the computational results of the expected makespan objective are reported in the section 5; Finally, Section 6 concludes this work.

2 Definition of Stochastic RCPSP

Before the stochastic RCPSP is examined, it is necessary to be familiar with the resource constrained project scheduling problem. RCPSP has $n+2$ activities and m renewable resources. The set of activities is denoted by $N = \{1,2,\dots,n+2\}$. The activities 1 and $n+2$ are called dummy activities representing the start and end activities, respectively. The duration of the activity i is d_i. There are different types of renewable resources, and each resource has a limited quantity. The availability of each resource of Type k is R_k units, $k = 1,2,\dots,K$ (K is the number of different resources); the activity i requires r_{ik} units of the resource k. It is assumed that A is an order relation on N such that $(i,j) \in A$, if and only if the activity j cannot start before the activity i has finished; A is called a precedence relation. A feasible schedule for the project is represented by $S = (S_1, S_1, \dots, S_{n+2})$ where S_i is the starting time of the activity i. The completion of the project is S_{n+2} (Subsequently referred to as C_{max}). RCPSP is modeled as follows:

$$\begin{aligned} &\text{Minimize} && C_{max} \\ &\text{subject to} && \\ &&& S_i \le S_j - d_i && \forall (i,j) \in A \\ &&& \sum_{i \in P(s,t)} r_{ik} \le R_k && k \in R, t \in [0, S_{n+2}] \end{aligned} \tag{1}$$

where $p(s,t)$ represents the set of activities in process at time t [16].

Stochastic RCPSP is a stochastic variant of RCPSP in which the activity durations will be random variables with a specific probability distribution. The random vector $D = (d_1, d_2, \dots, d_{n+2})$ shows the activity durations. For the start and end activities we have $P_r[D_i = 0] = 1$, and For the *i*th activity $i \in \{2, 3, \dots, n+1\}$ it is assumed that $P_r[D_i \le 0] = 0$, where $P_r[e]$ represents the probability of event e.

For the stochastic RCPSP the objective is to minimize the expected makespan. The expected makespan of a priority list λ is computed using the following equation,

$$E[\lambda, nscen] = \frac{1}{nscen} \sum_{d \in D} C_{max}\left(S^{\lambda}(d)\right), \tag{2}$$

where $nscen$ is the number of different scenarios, D is a sample of $nscen$ independent scenarios of *d*, and $S^{\lambda}(d) = (S_1^{\lambda}(d), \dots, S_{n+2}^{\lambda}(d))$ is the schedule that will be obtained by applying the stochastic Serial-SGS on the priority list λ under the scenario d. Finally $C_{max}\left(S^{\lambda}(d)\right)$ is equal to $S_{n+2}^{\lambda}(d)$.

3 Artificial Bee Colony (ABC) Algorithm

In the past decade, metaheuristics have been successfully applied on the NP-hard optimization problems. Due to NP-hardness of stochastic RCPSP, the metaheuristics such as GA and GRASP have been used by authors to solve it. In this paper, the artificial bee colony is used to solve this problem. The ABC algorithm is one of the newest and most successful metaheuristics presented to solve optimization problems. The behavior of honey bees has been studied extensively by natural scientists and formulated with the model of bee′s behavior. The ABC algorithm has been inspired by the intelligent behavior of real honey bees [10], [11].

One of the examples of bee colony behavior is the waggle dance of bees during the food procuring. When a worker honey bee has visited a food resource, she returns to her hive, and she performs a waggle dance on the vertical face of the honey comb to inform the other bees in the hive about the location of the food source. This behavior can cause additional workers to move to the location, thus enabling the colony to exploit the food source effectively.

In this paper, the artificial bee colony consists of three groups of bees: Employed, onlooker and scout bees. The first half of the colony consists of the employed artificial bees and the second half constitutes the onlooker bees. For every food source there is only one employed bee. The employed bee whose food source has been exhausted becomes a scout.

In a robust search, the exploration process and the exploitation process must be carried out simultaneously. In the ABC, while the onlooker bees and employed bees carry out the exploitation process in the search space, the scout controls the exploration

process. These three steps are repeated until the termination criteria are satisfied [10]. The search carried out by the artificial bees can be summarized as follows:

Such employed bee determines a food source in the neighborhood of the food source in her memory. Employed bees share their information with the onlooker bees waiting in the hive and then the onlooker bees select one of the food sources advertized by the employed bees. An onlooker bee moves toward the chosen food source in the previous step and selects a new food source in the neighborhood of the current position. A bee whose food source has been abandoned becomes a scout and starts to search a new food source randomly. The primary food sources are determined randomly. At the update stage, an employed bee will choose a feasible solution (food source) in the neighborhood of its current position, randomly and moves toward a new position. If the new food source has better nectar, the employed bee will change its position. The employed bees in the neighborhood of the current situation move according to the following motion equation:

$$x_{id}^{new} = x_{id}^{old} + x_{id}\left(x_{id}^{old} - x_{kd}\right) \quad (3),$$

in this equation, x_k is a position in the neighborhood of the employed bee i and selected randomly. The vector x_i^{old} shows the bee's previous status. The vector Φ_i is a random vector so that each component of this vector will be located between -1 and 1. After the employed bees have explored the new areas of food sources, they will come into the hive and share the information with the onlooker bees. Now an onlooker bee needs a process to select an employed bee as her guide. For this purpose, the probability for the employed bee *i* will be calculated as follows,

$$P_i = \frac{f_i}{\sum_{n=1}^{SN} f_n} \quad (4),$$

where f_i for the maximization problems is equal to $fitness_i$ and for the minimization problems is calculated according to the following equation,

$$f_i = \frac{1}{fitness_i} \quad (5),$$

where $fitness_i$ is the objective function value for the employed bee *i*. After calculating the probabilities, each onlooker bee employs the roulette wheel to choose an employed bee based on its probability. After selecting a guide, the onlooker bee updates its position using the following equation:

$$x_{id}^{new} = x_{id}^{old} + \phi_{id}\left(x_{id}^{old} - x_{kd}\right) \quad (6),$$

where x_k is the position of the employed bee that has been selected as a guide. Should the newly obtained position be better than the old one, the new position will replace the previous position. Now the food sources with poor qualities are abandoned and their associated bees become scouts. In the next section the way of solving a stochastic RCPSP by means of the ABC algorithm is presented.

4 Application of ABC Algorithm for Stochastic RCPSP

The details about the ABC algorithm to solve the stochastic RCPSP are expressed in this section. Figure 1 presents the proposed ABC algorithm to solve the stochastic RCPSP in pseudo code.

ABC Algorithm (*Population size, Num_scouts, Max-Numerator, D*)

```
  Calculate the average project.
  Calculate nscen independent scenarios from d.
  Initialization step
       Define SN = Population size/ 2
       For i = 1 to SN
              Initialize Position_i randomly
              Numerator_i= 0
       End For
  Evaluation step
       For i = 1 to SN
             EvaluationFunction(Position_i)
        End For
        population = {λ_1, λ_2, … , λ_SN}
  Updating step
  While termination criteria not met do
     (Send Employed Bees)
         For i = 1 to SN
               Select Neighborhood position randomly
               Calculat position_i^new by use of equation 3
               EvaluationFunction(position_i^new)
               If   ( Fitness (λ_i^new) < Fitness (λ_i))
                  population = (population \ λ_i) ∪ λ_i^new
               Else
                    Numerator_i = Numerator_i + 1
         End For
     (Send Onlooker bees)
         Calculate probabilities for each food source
using equation 4
         For i = 1 to SN
               Select Neighborhood k from food sources
based on roulette wheel
               Calculat position_i^new by use of equation 6
               EvaluationFunction(position_i^new)
               If   ( Fitness (λ_i^new) < Fitness (λ_i))
                  population = (population \ λ_i) ∪ λ_i^new
               Else
                    Numerator_i = Numerator_i + 1
         End For
     (Send Scout Bees)
         For i = 1 to SN
               If   Numerator_i > Max_Numerator
                    Initialize food source i randomly
```

Fig. 1. Pseudo code of the ABC algorithm for stochastic RCPSP

```
                    Numerator_i = 0
                End If
            End For
        End while
        Termination step
        Return best priority list.
```

Fig. 1. (*Continued*)

In the ABC algorithm (see Fig.1), to provide a priority list for a stochastic project and its corresponding food source, the function *EvaluationFunction* is used. This function is presented in the figure 2.

```
EvaluationFunction(Position_i)
        Evaluate priority list δ_i (for average project)
        Evaluate food source for average project using serial-SGS
        Evaluate priority list λ_i using LFT rule on the average project (for stochastic project)
        Evaluate food source for stochastic projects using stochastic serial-SGS
        Fitness (λ_i) = E[λ_i, nscen].
```

Fig. 2. Pseudo code of the EvaluationFunction for $Position_i$

At first, the function *EvaluationFunction* evaluates the priority list δ_i for an average project according to the position of the bee i; the average project is a deterministic project obtained from the stochastic project, where the duration of each activity is the mean of its duration distribution. Then, the serial schedule generation scheme (serial-SGS) is used to build a scheduling from the priority list δ_i [12]. The serial-SGS takes the activities one at a time from the priority list and schedules them at the earliest feasible precedence and resources. Next, a priority list λ_i is made using the LFT rule on the average project. To evaluate the food source i for the stochastic project, another type of policy is needed. In this paper, the stochastic serial-SGS is used to generate a feasible schedule. For a given sampled, the stochastic serial-SGS takes the activities one by one from the list and schedules them at the earliest possible time, but without overtaking the activities already scheduled. Finally, according to the equation 4 the fitness of the priority list λ_i is calculated.

The ABC algorithm consists of a set of different kinds of bees that find the priority list with the minimum expected makespan. The method used by a bee to find the food source specifies its type. A bee which finds a new food source simply according to her past position is called an employed bee. A bee that remains in the hive to detect a new food source by use of information from the employed bees is named as onlooker bee, and the bee which moves in the search space randomly is called a scout. The ABC algorithm uses the following steps to find a priority list with the minimum expected makespan:

(Initialization step) the ABC algorithm receives the following parameters as inputs: population size (*Population size*), the number of scouts (*Num_Scouts*), *Max_Numerator*, and D. Parameter D is the sample of *nscen* independent scenarios of d. *Max_Numerator* is the parameter utilized to denote the food sources that must to be abandoned.

At this step, the population is divided into two equal subsets that consist of the employed and onlooker bees. The number of employed bees (*SN*) will be set to half of the population size. $Numerator_i$ is the parameter that be increased when a food source is not developed in two continuous cycles. At last, the average project is created.

(Evaluation step). The initial population for the ABC algorithm is produced at the evaluation step. The function *EvaluationFunction* generates every member of the initial population and evaluates its corresponding food sources. The preliminary population is consists of all the priority lists obtained for the stochastic project.

(Updating Step). After the initial position of each bee has been identified, the employed bee i chooses a new position as its own neighborhood and moves according to the equation 3. According to the new position of the employed bee i ($position_i^{new}$), by using the function *EvaluationFunction*, a new priority list is made and the new food source of this bee for the stochastic project is obtained. After the employed bees have found new food sources and updated their current positions, they will return to the hive and share information about their food sources with the onlooker bees. By using the roulette wheel, each onlooker bee selects an employed bee k based on its probability (the equations 4 and 5). According to the information related to the employed bee food source, the onlooker bee updates its position using the equation 6. By using the function *EvaluationFunction*, a new priority list is generated and the new priority list is calculated. Should the newly discovered food source propose a priority list with a smaller expected makespan, the onlooker bee leaves her food source and goes to the new food source.

At each cycle of the ABC algorithm, the situations are assessed and should a food source not be able to be optimized after the *Max_Numerator* iteration, the corresponding food source is abandoned. The food source is replaced with the new one found by the scouts. In this paper the *Max_Numerator* is set to 10.

(Termination step). By the termination of the ABC algorithm, the priority list with the minimum expected makespan obtained by the population is returned as the output.

5 Computational Results

In this section, the results of our computational experiments to investigate the performance of the ABC algorithm and the other algorithms on the stochastic RCPSP are represented. All experiments were performed on a personal computer with 2.6 GHz CPU and 2.00 GB RAM. The coding was performed in C++. Our tests are performed on the instances from the benchmark library PSPLIB, which were generated by the problem generator ProGen [13].

Each instance consists of 122 activities and 4 different resources. The duration of activity i *(*$2 \leq i \leq 121$*)* is located between 1 and 10 *(i.e.* $2 \leq d_i^* \leq 121$*)*. The number

of instances are 600. The deterministic processing time d_i^* of each activity is taken as expectation of distribution. The uniform and exponential distributions are examined in order to choose the probability distributions. Three distributions were used: first of all, a continuous uniform distribution with support $[d_i - \sqrt{d_i}, d_i + \sqrt{d_i}]$ *('U1');* secondly, another continuous Uniform distribution with support $[0, 2d_i]$ *('U2');* finally, one exponential distribution with expectation d_i *('Exp').* The variance of these distributions are d_i, ${d_i}^2/3$ and ${d_i}^2$, respectively.

The quality of the algorithm is investigated by the bottom equation,

$$\text{pde} = \frac{1}{600}\sum_{\text{instances}}\left(\frac{\text{E}[\lambda,\text{nscen}]-\text{CPL}}{\text{CPL}}\right)\times 100, \quad (7)$$

where *CPL* is the critical-path length of the average project; pde is the average of the percentage distance of $\text{E}[\lambda, \text{nscen}]$ from the critical-path length of the average project.

In order to compare the algorithms used on the stochastic RCPSP more effectively, many limitations are considered. Limiting the number of schedules for the average project is one of these limitations. In this article, there are both 5000 and 25000 restrictions on the number of schedules.

Many scenarios for obtaining suitable approximation for the expected makespan are used. With a fixed number of schedules and the changing of the number of scenarios, the effect of this parameter on the quality of solution is investigated. The following experiments to test whether reducing the number of scenarios improves the quality of the solutions have been implemented. We have run ABC+S (ABC algorithm + Serial-SGS) with *nscen* being 5, 10, 20, 50, and 100 scenarios. Table 1contains the results of ABC+S with respect to the different number of scenarios for distributions $U1$, $U2$ and Exp, respectively.

As anticipated, a direct interface exists between the variability of the distribution and the fitness of the stochastic project; however, even with a very big variability, such as in Exp, a definite improvement can be attained. The improvement is dependent on the number of schedules generated for the average project.

Table 1. Changing the number of scenarios in ABC+S for $U1$, $U2$ and Exp (Percentage distance of $\text{E}[\lambda, \text{nscen}]$ from the critical-path length of the average project)

Distribution	$\mathbf{U(d_i \pm \sqrt{d_i})}$		$\mathbf{U(0, 2d_i)}$		$\mathbf{Exp(d_i)}$	
# schedules	**5000**	**25000**	**5000**	**25000**	**5000**	**25000**
5	54.28	51.85	74.86	71.48	100.75	**94.37**
10	**54.23**	51.92	74.60	71.41	**100.39**	95.74
20	54.46	**51.77**	**74.58**	**71.19**	100.56	95.12
50	54.44	51.90	74.81	71.42	100.87	96.23
100	54.39	51.81	74.77	71.26	100.61	95.60

According to the above-mentioned table, there is no number of scenarios for which the best results for all tests are obtained. For U1, 10 for 5000 and 20 for 25000 are the best choices; and for $U2$, 20 works best. For Exp, 10 is the best choice for 5000 and 5 works best for 25000.

Now we can compare the ABC algorithm with the other stochastic RCPSP-algorithms from the literature. The genetic algorithm (GA) and the GRASP algorithm are considered [2], [3], where the same data set and schedule restriction are used, with the distributions $U1$, $U2$ and Exp. The ABC algorithm performance is compared with the other algorithms for the problems with the distributions U1, U2 and Exp. The computation results are given in the table 2.

Table 2. Compare the ABC algorithm with the GA and GRASP for $U1$, $U2$ and Exp.

BEST	$\mathbf{U(d_i \pm \sqrt{d_i})}$		$\mathbf{U(0, 2d_i)}$		$\mathbf{Exp(d_i)}$	
	5000	**25000**	**5000**	**25000**	**5000**	**25000**
GA+S	52.14%	49.63%	78.65 %	75.38%	120.22 %	116.83%
GA+P	51.94%	50.16%	78.50%	76.33%	120.91%	118.56%
GRASP	**46.84%**	**45.21%**	**72.58 %**	**70.95%**	114.42 %	112.37%
ABC+S	54.23%	51.77%	74.58%	71.19%	**100.39%**	**95.12%**

As can be seen from the results, the GA and GRASP have better performance compared to ABC algorithm for the stochastic projects with the distribution $U1$ while for the stochastic projects with the distributions $U2$ and Exp the ABC algorithm has better performance than the genetic algorithm. The ABC algorithm shows better efficiency than GRASP when the variability is very large (Exp) (table 2).

6 Conclusions

In this paper, the ABC algorithm to solve the stochastic resource constrained project scheduling problem was used. The ABC algorithm begins with the initial priority lists and attempts to reduce the expected makespan through searching the feasible solutions space. The computational experiment showed that this algorithm produces good quality solutions. By comparing the numerical results, we can say that the ABC algorithm provides an efficient way to solve the stochastic RCPSP.

When the availability of distribution is large, the efficiency of the ABC algorithm is better than the other stochastic-algorithms in the literature. For example, in the stochastic problems with the exponential distribution, the *pde* of the proposed ABC algorithm over the test problems with 5000 schedules is 100.39 whereas for the GRASP and GA, the pdes are 114.42 and 120.22, respectively. Similarly, the pdes for ABC, GRASP and GA over test problems with 25000 schedules are 95.12, 112.37 and 116.83, respectively. Due to the ability of ABC algorithm in providing better diversity throughout the execution of the algorithm, the qualities of solutions are improved. Providing appropriate level of diversity helps the algorithm to alleviate the deficiencies of meta-heuristic algorithm such as stagnation and premature convergence and accordingly provide the ability to explore further regions of the search space to find better solutions.

Table 1shows that should the number of schedules to be generated for the average project is increased; the quality of the solution to the stochastic project will increase too.

References

1. Akbari, R., Zeighami, V., Ziarati, K.: Artificial Bee colony for resource constrained project scheduling problem. International Journal of Industrial Engineering Computations 1, 45–60 (2011)
2. Ballestin, F.: When it is worthwhile to work with the stochastic RCPSP? Journal of Scheduling 10, 153–166 (2007)
3. Ballestin, F., Leus, R.: Resource-constrained project scheduling for timely project completion with stochastic activity durations. Production and Operations Management 18, 459–474 (2009)
4. Blazewicz, J., Lenstra, J., Rinnooy Kan, A.: Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 11–24 (1983)
5. Gerchak, Y.: On the allocation of uncertainty-reduction effort to minimize total variability. IIE Transactions 32, 403–407 (2000)
6. Golenko-Ginzburg, D., Gonik, D.: Stochastic network project scheduling with non-consumable limited resources. International Journal of Production Economics 48, 29–37 (1997)
7. Gutjahr, W.J., Straus, C., Wagner, E.: A stochastic branch-and-bound approach to activity crashing in project management. INFORMS Journal on Computing 12, 125–135 (2000)
8. Hartmann, S., Kolisch, R.: Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 127, 394–407 (2000)
9. Igelmund, G., Radermacher, F.J.: Preselective strategies for the optimization of stochastic project networks under resource constraints. Networks 13, 1–28 (1983)
10. Karaboga, D.: An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
11. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 39, 459–471 (2007)
12. Kolisch, R., Hartmann, S.: Heuristic Algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In: Handbook of Recent Advances in Project Scheduling, pp. 147–178. Kluwer Academic Publishers, Boston
13. Kolisch, R., Sprecher, A.: PSPLIB - a project scheduling problem library. European Journal of Operational Research 96, 205–216 (1996)
14. Stork F.: Stochastic resource-constrained project scheduling, Ph.D. thesis, Technische UniversitÄat Berlin, (2001)
15. Tsai, Y.-W., Gemmill, D.D.: Using tabu search to schedule activities of stochastic resource-constrained projects. European Journal of Operational Research 111, 129–141 (1998)
16. Valls, V., Ballestın, F., Quintanilla, S.: A hybrid genetic algorithm for the resource constrained project scheduling problem. European Journal of Operational Research 185, 495–508 (2008)
17. Wollmer, R.D.: Critical path planning under uncertainty. Mathematical Programming Study 25, 164–171 (1985)

Brain Storm Optimization Algorithm

Yuhui Shi

Xi'an Jiaotong-Liverpool University
Suzhou, China 215123
yuhui.shi@xjtlu.edu.cn

Abstract. Human being is the most intelligent animal in this world. Intuitively, optimization algorithm inspired by human being creative problem solving process should be superior to the optimization algorithms inspired by collective behavior of insects like ants, bee, *etc*. In this paper, we introduce a novel brain storm optimization algorithm, which was inspired by the human brainstorming process. Two benchmark functions were tested to validate the effectiveness and usefulness of the proposed algorithm.

Keywords: Optimization, Brainstorming Process, Brain Storm Optimization.

1 Introduction

Population-based optimization algorithms have been widely accepted and successfully applied to solve a lot of optimization problems. Unlike traditional single-point based algorithms such as hill-climbing algorithms, a population-based optimization algorithm consists of a set of points (population) which solve the problem through information sharing to cooperate and/or compete among themselves. So far, there are a lot of population-based algorithms existed. The very first population-based algorithms are evolutionary algorithms including evolutionary programming, genetic algorithm, evolution strategy, and genetic programming [EBERHART2007], which were inspired by biological evolution. Recently, there occurred more population-based algorithms which are usually called nature-inspired optimization algorithms instead of evolution-inspired algorithms. Many of nature-inspired optimization algorithms are categorized as swarm intelligence algorithms. In a swarm intelligence algorithm, each individual in the population represents a simple object such as ant, bird, fish, *etc*. There exist a lot of different swarm intelligence algorithms, among which are particle swarm optimization (PSO) [SHI1998], ant colony optimization algorithm(ACO) [DORIGO1996], bacterial forging optimization algorithm(BFO) [PASSINO2010], firefly optimization algorithm [YANG2008], artificial immune system [CASTRO1999], and *etc*.

In a swarm intelligence algorithm, it is the collective behavior of all individuals that makes the algorithm to be effective in problem optimization. All individuals cooperate and collectively move toward the better and better areas in the solution search space. These individuals represent only simple objects such as birds in PSO, ants in ACO, bacteria in BFO, *etc*. Human beings are social animals and are the most intelligent

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 303–309, 2011.
© Springer-Verlag Berlin Heidelberg 2011

animals in the world. Therefore, it is natural to expect that an optimization algorithm inspired by human creative problem solving process will be a good optimization algorithm. In this paper, we will introduce a novel optimization algorithm inspired by the human idea generation process – brainstorming process.

The remaining paper is organized as follows. In Section 2, the human brainstorming process is reviewed. In Section 3, the novel optimization algorithm inspired by human brainstorming process is introduced and described in detail, followed by experimental simulation and result discussion on two benchmark functions in Section 4. Finally, conclusions are given in Section 5.

2 Brainstorming Process

As we all may have experienced that when we face a difficult problem which every single person can't solve, a group of persons, especially with different background, get together to brain storm, the problem can usually be solved with high probability. Great and un-expectable intelligence can occur from interactive collaboration of human beings. One way to help human beings to interactively collaborate to generate great ideas is to get together a group of people to brainstorm. A brainstorming process generally follows the steps listed in Table 1.

Table 1. Steps in a Brainstorming Process

Step 1.	Get together a brainstorming group of people with as diverse background as possible;
Step 2.	Generate many ideas according to the rules in Table 2;
Step 3.	Have several, say 3 or 5, clients act as the owners of the problem to pick up several, say one from each owner, ideas as better ideas for solving the problem;
Step 4.	Use the ideas picked up in the Step 3 with higher probability than other ideas as clues, and generate more ideas according to the rules in Table 2;
Step 5.	Have the owners to pick up several better ideas generated as did in Step 3;
Step 6.	Randomly pick an object and use the functions and appearance of the object as clues, generate more ideas according to the rules in Table 2;
Step 7.	Have the owners to pick up several better ideas;
Step 8.	Hopefully a good enough solution can be obtained by considering and/or merging the ideas generated.

In a brainstorming process, usually there are a facilitator, a brainstorming group of people, and several owners of the problem to be solved. The role of the facilitator is to facilitate the idea generation (brainstorming) process by enforcing the brainstorming group to obey the Osborn's original four rules of idea generation in a brainstorming process [SMITH2002]. The four rules are listed in Table 2 below. The facilitator should not be involved in generating ideas itself, but facilitating the brainstorming process only. The guideline for selecting facilitator is to have a facilitator to have facilitation experience but have less expertise on the background knowledge related to the problem to be solved as possible. The purpose of this is to have generated ideas to have less, if not none, biases from the facilitator.

Table 2. Osborn's Original Rules for Idea Generation in a Brainstorming Process

Rule 1.	Suspend Judgment	Rule 2.	Anything Goes
Rule 3.	Cross-fertilize (Piggyback)	Rule 4.	Go for Quantity

In Table 2, the Rule 1 says that there is no idea as bad idea. All ideas are good ideas. It is unwise to judge whether a proposed idea is a good or bad idea. Any judgment or criticism must be held back until at least the end of the brainstorming process. The Rule 2 says that anything coming to your mind during the brainstorming process is an idea worth to be shared and recorded. Don't let any idea or thought ignored. The Rule 3 says that lot of ideas can and should be based on ideas already generated. Any generated idea can and should serve as a clue to generate more ideas. The Rule 4 says that it is necessary to generate as many ideas as possible. We first go for quantity of generated ideas. The quality will come from quantity naturally. Without generating large quantity of ideas, it is difficult, if not impossible, to come out ideas with good quality.

The purpose to generate ideas according to rules in Table 2 is to generate ideas as diverse as possible so that the people in the brainstorming group will be open-minded as much as possible. The operation of picking up an object in Step 6 serves for the same purpose as generating diverse and different ideas. It can help brainstorming group to diverge from previously generated ideas therefore to avoid being trapped by the previously generated ideas. As a consequence, the brainstorming group will be more open-minded and generate more diverse ideas. The problem owners serve for one different purpose. Picking up several good ideas from ideas generated so far is to cause the brainstorming group to pay more attention to the better ideas which the brainstorming group believes to be. The ideas picked-up work like point-attractors for the idea generation process.

3 Brain Storm Optimization Algorithm

The brainstorming process has been successfully applied to generate ideas to solve very difficult and challenging problems. Intuitively, an optimization algorithm designed based on the human being idea generation process should be superior to optimization algorithms inspired by collective behaviors of animals because human beings are the most intelligent animals in the world. The novel optimization algorithm inspired by brainstorming process is given in Table 3.

In the procedure of the Brain Storm Optimization (BSO) algorithm shown in the Table 3, the Step 1 is the initialization step as that in other population-based algorithms; the Step 2, 3, and 4 in Table 3 serves the purpose of Step 3, 5, and 7 in Table 1 to pick up several better ideas; the Step 5 in Table 3 simulates the operation of Step 6 in Table 1 to make the population to cover unexplored areas; the Step 6 in Table 3 simulates the idea generation in Step 2, 4, and 6 of Table 1; the Step 6.b simulates generating new idea inspired by one single existing idea while the Step 6.c simulates generating new

Table 3. Procedure of Brain Storm Optimization Algorithm

1. Randomly generate n potential solutions (individuals);
2. Cluster n individuals into m clusters;
3. Evaluate the n individuals;
4. Rank individuals in each cluster and record the best individual as cluster center in each cluster;
5. Randomly generate a value between 0 and 1;
   a) If the value is smaller than a pre-determined probability $p_{5a,}$
      i. Randomly select a cluster center;
      ii. Randomly generate an individual to replace the selected cluster center;
6. Generate new individuals
   a) Randomly generate a value between 0 and 1;
   b) If the value is less than a probability p_{6b},
      i. Randomly select a cluster with a probability p_{6bi};
      ii. Generate a random value between 0 and 1;
      iii. If the value is smaller than a pre-determined probability p_{6biii},
         1) Select the cluster center and add random values to it to generate new individual.
      iv. Otherwise randomly select an individual from this cluster and add random value to the individual to generate new individual.
   c) Otherwise randomly select two clusters to generate new individual
      i. Generate a random value;
      ii. If it is less than a pre-determined probability p_{6c}, the two cluster centers are combined and then added with random values to generate new individual;
      iii. Otherwise, two individuals from each selected cluster are randomly selected to be combined and added with random values to generate new individual.
   d) The newly generated individual is compared with the existing individual with the same individual index, the better one is kept and recorded as the new individual;
7. If n new individuals have been generated, go to step 8; otherwise go to step 6;
8. Terminate if pre-determined maximum number of iterations has been reached; otherwise go to step 2.

idea inspired by two existing ideas from two different idea clusters, respectively; the cluster center has more chances to be used to generate new ideas than other ideas in each cluster; certainly a new idea can also be inspired by more than two existing ideas, but it is not simulated in the BSO algorithm for keeping the algorithm simple; the number of clusters in Table 3 serves as the role of problem owners in Table 1; the cluster center in each cluster serves the purpose of better ideas picked up by problem owners. The Step 6.d serves the purpose of keeping better ideas generated. The population size n simulates the number of ideas generated in each round of idea generation in the brainstorming process. For the simplicity of the algorithm, the

population size usually is set to be a constant number for all iterations in the BSO algorithm.

4 Experiments and Discussions

The BSO procedure can be implemented in various ways by setting up BSO algorithm's parameters differently. In Step 6.b.i of the BSO procedure shown in Table 3, a cluster is selected with probability p_{6bi}, which is proportional to the number of individuals in the cluster. This is, the more individuals a cluster contains, the more likely it will be selected. The Gaussian random values will be used as random values which are added to generate new individuals. The new individual generation in Step 6 can be represented as

$$X_{new}^{d} = X_{selected}^{d} + \xi * n(\mu, \sigma) \quad (1)$$

where $X_{selected}^{d}$ is the d^{th} dimension of the individual selected to generate new individual; X_{new}^{d} is the d^{th} dimension of the individual newly generated; $n(\mu, \sigma)$ is the Gaussian random function with mean μ and variance σ; the ξ is a coefficient that weights the contribution of the Gaussian random value. For the simplicity and for the purpose offine tuning, the ξ can be calculated as

$$\xi = \text{logsig}((0.5 * \text{max_iternation} - \text{current_iteration})/k) * \text{rand}() \quad (2)$$

where logsig() is a logarithmic sigmoid transfer function, max_iteration is the maximum number of iterations, and current_iteration is the current iteration number, k is for changing logsig() function's slope, and rand() is a random value within (0,1).

For the purpose of validating the effectiveness and usefulness of the proposed BSO algorithm, in this paper, a set of parameters are set intuitively for the procedure of BSO given in Table 3 for both testing functions below. The set of parameters are listed in Table 4 below.

Table 4. Set of Parameters for BSO Algorithm

n	m	p_{5a}	p_{6b}	p_{6biii}	p_{6c}	k	Max_ iteration	μ	σ
100	5	0.2	0.8	0.4	0.5	20	2000	0	1

The BSO algorithm is then tested on two benchmark functions listed in Table 5. We use k-mean cluster algorithm to cluster n individuals into m clusters. For each benchmark function, the BSO will be run 50 times to obtain reasonable statistical results.

The first benchmark function is the Sphere function, which is an unimodal function. The simulation results of testing BSO on the Sphere function with dimensions 10, 20, and 30, respectively, are given in Table 6.

Table 5. Benchmark Functions Tested in This Paper

Index	Function name	Function expression
1	Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$
2	Rastrigin	$f_2 = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$

Table 6. Simulation Results of BSO Algorithm

function	dimension	mean	best	worst	variance
Sphere	10	3.82E-44	1.50775E-44	7.12557E-44	1.57592E-88
	20	3.1E-43	1.61402E-43	4.56276E-43	4.0471E-87
	30	1.15E-42	8.07001E-43	1.69603E-42	4.69513E-86
Rastrigin	10	3.820643	1.989918	6.964713	1.954026
	20	18.06844	8.954632	26.86387	19.65172
	30	32.91322	17.90926	58.70249	82.82522

The results given in the Table 6 are mean, best, worst minimum values and their variance of the final iteration over 50 runs. From the Table 6, it can be observed that very good results can be obtained by the implemented BSO algorithm.

The second is the Rastrigin function, which is a multimodal function. The simulation results of testing BSO on the Rastrigin function with dimensions 10, 20, and 30, respectively, are given in Table 6. From Table 6, we can observe that good results can also be obtained by this very version of BSO algorithm.

5 Conclusions

In this paper, we introduced a novel brain storm optimization algorithm which was inspired by the human brainstorming process. Human beings are the most intelligent animals in the world, therefore, it is natural to believe that the optimization algorithm inspired by collective behavior of human beings should be superior to the optimization algorithms inspired by collective behavior of injects such as ants, birds, *etc*. The proposed BSO algorithm was implemented and tested on two benchmark functions. Even though the simulation results are very preliminary, the results DID validate the effectiveness and usefulness of the proposed BSO algorithm which is the purpose of this paper. More detailed analysis and experimental tests are our next step research work, e.g. using different clustering algorithm, using different random functions for generating new individuals, *etc*.

Acknowledgement

This paper is partially supported by National Natural Science Foundation of China under Grant No.60975080, and by Suzhou Science and Technology Project under Grant No.SYJG0919.

References

1. de Castro, L.N., Von Zuben, F.J.: Artificial Immune Systems: Part I -Basic Theory and Applications, School of Computing and Electrical Engineering, State University of Campinas, Brazil, No. DCA-RT 01/99 (1999)
2. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst., Man, Cybern. B 26(2), 29–41 (1996)
3. Eberhart, R.C., Shi, Y.: Computational Intelligence, Concepts to Implementation, 1st edn. Morgan Kaufmann Publishers, San Francisco (2007)
4. Passino, K.M.: Bacterial Foraging Optimization. International Journal of Swarm Intelligence Research 1(1), 1–16 (2010)
5. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, May 4-9 (1998)
6. Smith, R.: The 7 Levels of Change, 2nd edn. Tapeslry Press (2002)
7. Yang, X.: Nature-Inspired Metaheuristic Algorithms. Luniver Press (2008)

Human Group Optimizer with Local Search

Chaohua Dai, Weirong Chen, Lili Ran, Yi Zhang, and Yu Du

The School of Electrical Engineering, Southwest Jiaotong University,
610031 Chengdu, China
dchzyf@yahoo.com.cn

Abstract. Human Group Optimization (HGO) algorithm, derived from the previously proposed seeker optimization algorithm (SOA), is a novel swarm intelligence algorithm by simulating human behaviors, especially human searching/foraging behaviors. In this paper, a canonical HGO with local search (L-HGO) is proposed. Based on the benchmark functions provided by CEC2005, the proposed algorithm is compared with several versions of differential evolution (DE) algorithms, particle swarm optimization (PSO) algorithms and covariance matrix adaptation evolution strategy (CMA-ES). The simulation results show that the proposed HGO is competitive or, even, superior to the considered other algorithms for some employed functions.

Keywords: Human group optimizer, human searching/foraging behaviors, seeker optimization algorithm, swarm intelligence.

1 Introduction

During the past over two decades, swarm intelligence (SI) [1-3], illumined by the social behavior of gregarious insects and other animals, has been attracting more and more attention of researchers. In the existing SI algorithms, researchers have focused mainly on the social behaviors of non-human animals [4], yet few have taken into account human social behaviors, especially human group searching/foraging behaviors. However, interacting groups of people also create emergent self-organizing behaviors that are not intended by any person [5], and swarm intelligence is known as an ever-present potential in humans [4]. As we all know, optimization tasks are often encountered in many areas of human life, and the search for a solution to a problem is one of the basic behaviors to all mankind [6]. In addition, in the search process, human brain can effectively manage the trade-off between exploitation and exploration [7]. Hence, it is an interesting choice to simulate human behaviors, especially human group searching/foraging behaviors, for solving optimization problems. The seeker optimization algorithm (SOA) [8] proposed by the authors of this paper is just such a paradigm. The SOA has been applied to function optimization [6], proton exchange membrane fuel cell model optimization [9], optimization problems in power systems [10-14], optimal assembly tolerance design [15], digital IIR filter design [16], and training neural networks [17], etc. According to these applications, it is preliminarily proved that the SOA is a promising and competitive candidate of heuristic search algorithms. Recently, because the term, seeker

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 310–320, 2011.
© Springer-Verlag Berlin Heidelberg 2011

optimization algorithm, could not reflect the essential nature of the novel algorithm of simulating ***human*** behaviors, it was renamed as Human Group Optimization algorithm or Human Group Optimizer (HGO) in Ref. [18]. At the same time, in Ref. [18], we proposed a canonical version of HGO. In this paper, to improve its local search ability, the Quasi-Newton method is combined, and the HGO with local search (L-HGO) is proposed. Furthermore, the performance of L-HGO on several benchmark functions provided by CEC2005 is reported.

The rest of this paper is organized as follows. In Section 2, we detail the HGO and L-HGO. Then, the L-HGO is compared with other algorithms by use of benchmark function optimization in Section 3. Finally, the conclusion is presented in Section 4.

2 Human Group Optimizer with Local Search

Human group optimization (HGO) algorithm operates on a set of solutions called search population, and the individual of this population is called seeker (i.e., a person). Assume that the optimization problems to be solved are minimization problems.

2.1 Implementation of Human Group Optimization

In HGO, a search direction $d_{ij}(t)$ and a step length $\alpha_{ij}(t)$ are computed separately for each seeker i on each dimension j for each time step t, where $\alpha_{ij}(t) \geq 0$ and $d_{ij}(t) \in \{-1,0,1\}$. For each seeker i ($1 \leq i \leq s$, s is the population size), the position update on each dimension j ($1 \leq j \leq D$) is given by (1):

$$x_{ij}(t+1) = x_{ij}(t) + \alpha_{ij}(t) d_{ij}(t) . \tag{1}$$

where $\vec{x}_i(t) = [x_{i1}, x_{i2}, \cdots, x_{iD}]$ is the D-dimensional position vector of the i-th seeker at time step t. The pseudocode of HGO is shown as Fig. 1.

```
Generating s positions randomly, t←0;
repeat
  Evaluating each seeker i;
  Selecting the neighbors for each seeker i;
  Computing d̄_i(t) and ᾱ_i(t) for each seeker i;
  Updating each seeker's position using (1);
  t←t+1;
until the stopping condition is satisfied.
```

Fig. 1. The pseudocode of the HGO

2.2 Search Direction

In HGO, each seeker i selects his search direction synthetically based on several empirical gradients (EGs) by evaluating the current or historical positions of himself or his neighbors. Generally, the EGs involve egotistic behavior, altruistic behavior

and pro-activeness behavior to yield an egotistic direction, two altruistic directions and a pro-activeness direction, respectively [11, 16].

$$\vec{d}_{i,\mathrm{ego}}(t) = \mathrm{sign}(\vec{p}_{i,\mathrm{best}}(t) - \vec{x}_i(t)). \tag{2}$$

$$\vec{d}_{i,\mathrm{alt}_1}(t) = \begin{cases} \mathrm{sign}(\vec{g}_{\mathrm{best}}(t) - \vec{x}_i(t)) & \text{if } \vec{g}_{\mathrm{best}} \text{ is better than } \vec{x}_i \\ \mathrm{sign}(\vec{x}_i(t) - \vec{g}_{\mathrm{best}}(t)) & \text{else} \end{cases} \tag{3}$$

$$\vec{d}_{i,\mathrm{alt}_2}(t) = \begin{cases} \mathrm{sign}(\vec{l}_{\mathrm{best}}(t) - \vec{x}_i(t)) & \text{if } \vec{l}_{\mathrm{best}} \text{ is better than } \vec{x}_i \\ \mathrm{sign}(\vec{x}_i(t) - \vec{l}_{\mathrm{best}}(t)) & \text{else} \end{cases} \tag{4}$$

$$\vec{d}_{i,\mathrm{pro}}(t) = \mathrm{sign}(\vec{x}_i(t_1) - \vec{x}_i(t_2)) \tag{5}$$

where $\vec{p}_{i,\mathrm{best}}(t)$ and $\vec{g}_{\mathrm{best}}(t)$ are the i-th seeker's and neighbors' historical best positions respectively, $\vec{l}_{\mathrm{best}}(t)$ is the neighbors' current best position, the $\mathrm{sign}(\cdot)$ is a signum function on each dimension of the input vector, and $\vec{x}_i(t_1)$ and $\vec{x}_i(t_2)$ are the best and the worst one from the set $\{\vec{x}_i(t), \vec{x}_i(t-1), \vec{x}_i(t-2)\}$, respectively.

According to human rational judgment, the actual search direction of the i-th seeker, $\vec{d}_i(t)$, is based on a compromise among the aforementioned four empirical directions applying the following proportional selection rule for each dimension j:

$$d_{ij} = \begin{cases} 0 & \text{if } r_j \le p_j^{(0)} \\ +1 & \text{if } p_j^{(0)} < r_j \le p_j^{(0)} + p_j^{(+1)} \\ -1 & \text{if } p_j^{(0)} + p_j^{(+1)} < r_j \le 1 \end{cases}. \tag{6}$$

where r_j is a uniformly random number in [0,1], $p_j^{(m)}$ ($m \in \{0,+1,-1\}$) is the percent of the number of "m" on each dimension j of the four empirical directions.

2.3 Step Length

In this algorithm, human focusing search is introduced and described by a simple Fuzzy control rule as "If {*fitness value* is *small*} (i.e., the *conditional* part), Then {*step length* is *short*}(i.e., the *action* part)". Hence, Fuzzy reasoning is used to determine step length of each seeker.

To design a Fuzzy system to be applicable to a wide range of optimization problems, the fitness values of all the seekers are descendingly sorted and turned into the sequence numbers from 1 to s as the inputs of Fuzzy reasoning. The linear membership function is used in the conditional part since the universe of discourse is a given set of numbers, i.e., $1,2,\cdots,s$. The expression is presented as (7).

$$\mu_i = \mu_{\max} - \frac{s - I_i}{s-1}(\mu_{\max} - \mu_{\min}). \tag{7}$$

where I_i is the sequence number of $\vec{x}_i(t)$ after sorting the fitness values, $\mu_{\max}$ and $\mu_{\min}$ are the maximum and minimum membership degree value. At the same time, the Bell membership function $\mu(x) = e^{-x^2/2\delta^2}$ is used in the action part. The parameter, $\vec{\delta}$, of the Bell membership function is determined by (8).

$$\vec{\delta} = \omega \cdot \text{abs}(\vec{x}_{\text{best}} - \vec{x}_{\text{rand}}) . \tag{8}$$

where $\text{abs}(\cdot)$ returns an output vector such that each element of the vector is the absolute value of the corresponding element of the input vector. The parameter ω is used to decrease the *step length* with time step increasing so as to gradually improve the search precision. The $\vec{x}_{\text{best}}$ and $\vec{x}_{\text{rand}}$ are the best seeker and a randomly selected neighbor from the population, respectively.

To introduce the randomicity on each dimension and improve local search capability, Eq. (9) is used to change μ_i in (7) into a vector $\vec{\mu}_i$. The *action* part of the Fuzzy reasoning gives every dimension *j* of *step length* by (10) where δ_j is the *j*-th dimension of the vector $\vec{\delta}$ in (8).

$$\mu_{ij} = RAND(\mu_i, 1) . \tag{9}$$

$$\alpha_{ij} = \delta_j \sqrt{-\ln(\mu_{ij})} \tag{10}$$

2.4 The Neighbors of Every Seeker

Before calculating the two altruistic directions in (3) and (4), each seeker needs to first give his neighborhood. Note: the seeker excludes himself from his neighborhood. In this work, a random topology introduced in [19, 20] is used, namely, each seeker randomly select other seekers as his neighbors with a selection probability:

$$p = 1 - (1 - \frac{1}{s})^K . \tag{11}$$

where *K* is an integer less than *s*.

2.5 Local Search Schedule

To our knowledge, unlike particle swarm optimization (PSO) who implements an aggregating search around the personal and neighbors' historical best positions, HGO utilizes a focusing search around the personal current positions. So, compared with PSO, HGO may more effectively keep the population diversity. However, we also found that the HGO sometimes lacks a sufficient local search ability to achieve a better search precision [18] although it can not only efficiently balance exploration and exploitation but also perfectly switch between a "nearer is better" assumption and a "nearer is worse" one [11, 16]. To give a better local search precision, a local search schedule introduced by Ref. [21] is added into HGO: every *L* (200 in this study)

generations, sort the population according to their fitness values and refine 5% individuals including the best individual and the randomly selected individuals out of the best 50% individuals in the current population using the Quasi-Newton method.

3 Simulation Results

In this section, the modified HGO is evaluated by using the first 14 functions of the test suite on real optimization of CEC 2005 [22]. Experiments are conducted on all the 10-dimension problems. The detailed evaluation criterion description can be referred to Ref. [22]. In this study, the parameters of the L-HGO are assigned as: $s = \text{int}(10 + 2 \cdot \sqrt{D})$, K=3, $\mu_{\max}$=0.95, $\mu_{\max}$=0.0111, and ω linearly decreased from 0.8 to 0.2 with time step increasing.

The proposed method is compared with self-adaptive differential evolution algorithm with local search (L-SaDE) [23], differential evolution with self-adapting control parameters (SACP-DE)[24], the restart version of the (μ_W,λ)-CMA-ES (LR-CMA-ES) [25], dynamic multi-swarm particle swarm optimizer with local search (DMS-L-PSO) [21], comprehensive learning particle swarm optimizer (CLPSO)[26] and Standard PSO 2007 (SPSO-07) [20]. The following results of L-SaDE, LR-CMA-ES, DMS-L-PSO and HGO are cited from the corresponding references.

For each problem, function error values achieved when *FEs*=1e+5 are tabulated in Table 1, successful *FEs* and success performance are presented in Table 2. Those successful *FEs* and Success Performance for functions 8, 13 and 14 are not shown because all the algorithms failed.

According to the simulation results in Table 1 and 2, the analysis on the comparisons between L-HGO and other algorithms is presented as follows. Except function 3 among the first five unimodal functions, L-HGO can successfully solve the other four functions and has the smallest or equal values of the *best*, *worst*, and *mean* function errors with 100% success rates. At the same time, L-HGO has comparable successful *FEs* and success performance from Table of Table 2 although it is not the best ones. Hence, L-HGO has good local search ability and rapid convergence rate. For function 3, L-HGO has a relatively bad performance.

For the multimodal functions from function 6 to function 12, L-HGO also presents good global search ability. L-HGO achieves 100% success rate for function 6 and 72% for function 7. For function 8, L-HGO along with all the other algorithms fails in all 25 runs. L-HGO can find the global optimum and achieves 52% success rate for function 9. Although L-HGO is not that good for function 10 owing to the rotation and only achieves 16% success rate, it can still find the precise global optimum. For function 11, the results of L-HGO are superior to all other algorithms with success rate of 36%, and L-HGO achieves 44% success rate for function 12. Functions 13 and 14 are extended functions and all the algorithms including L-HGO fail in all 25 runs for them. But, L-HGO has the comparable performance for function 13 and better function error values for function 14 than all the other algorithms.

Table 1. Function errors of various algorithms when *FEs*=1e+5

Funcs.		L-SaDE	SACP-DE	LR-CMA-ES	DMS-L-PSO	CLPSO	SPSO-2007	HGO	L-HGO
F1	1st	0	0	1.84e-9	0	0	0	0	0
	7th	0	0	3.75e-9	0	0	0	0	0
	13th	0	0	5.65e-9	0	0	0	0	0
	19th	0	0	6.42e-9	0	0	0	0	0
	25th	0	0	9.34e-9	0	0	0	0	0
	Mean	0	0	5.20e-9	0	0	0	0	0
	Std	0	0	1.94e-9	0	0	0	0	0
F2	1st	0	4.0168e-7	2.21e-9	0	0	5.6843e-14	0	0
	7th	0	1.0935e-6	3.27e-9	5.6843e-14	5.6843e-14	1.1369e-13	0	0
	13th	0	1.5977e-6	4.53e-9	5.6843e-14	5.6843e-14	1.7053e-13	0	0
	19th	0	2.7079e-6	5.71e-9	1.1369e-13	1.1369e-13	2.2737e-13	0	0
	25th	2.5580e-12	5.3480e-6	7.67e-9	7.3896e-13	1.7053e-13	4.5475e-13	5.6843e-14	5.6843e-14
	Mean	1.0459e-13	1.9829e-6	4.70e-9	1.2960e-13	7.9581e-14	1.9554e-13	2.2737e-15	1.1369e-14
	Std	5.1124e-13	1.4144e-6	1.56e-9	1.5612e-13	3.6692e-14	1.0126e-13	1.1369e-14	2.3206e-14
F3	1st	0	5.3347e+2	2.21e-9	1.5364e-9	4.7806e+3	3.3457e+3	8.1456e+3	1.2008e+2
	7th	0	1.5730e+3	4.61e-9	5.5371e-9	2.9139e+4	1.3958e+4	3.3641e+4	3.1296e+3
	13th	0	2.3763e+3	5.51e-9	7.3142e-9	5.5009e+4	3.0878e+4	5.6796e+4	1.2461e+4
	19th	9.9142e-6	3.3918e+3	6.58e-9	8.9301e-9	2.3547e+5	4.1905e+4	8.3709e+4	2.1470e+4
	25th	1.0309e-4	9.4322e+3	9.66e-9	1.1275e-8	1.4237e+6	1.0958e+5	2.3856e+5	1.0250e+5
	Mean	1.6720e-5	2.8819e+3	5.60e-9	7.0064e-9	2.2900e+5	3.3535e+4	6.8079e+4	1.8221e+4
	Std	3.1196e-5	2.2021e+3	1.93e-9	2.6589e-9	3.5814e+5	2.5853e+4	5.2735e+4	2.3743e+4
F4	1st	0	5.7618e-6	1.71e-9	2.0784e-4	5.6843e-14	2.1542e+0	0	0
	7th	0	1.6068e-5	3.85e-9	6.6984e-4	1.1369e-13	8.0694e+1	0	0
	13th	0	2.0628e-5	4.78e-9	1.1399e-3	4.5475e-13	2.4601e+2	0	0
	19th	0	4.0977e-5	6.46e-9	2.5429e-3	1.5348e-12	5.4506e+2	0	0
	25th	3.5456e-4	1.0043e-4	7.80e-9	8.7836e-3	9.0031e-9	2.7671e+3	5.6843e-14	1.1369e-13
	Mean	1.4182e-5	3.1071e-5	5.02e-9	1.8851e-3	4.0073e-10	5.3680e+2	9.0949e-15	2.9559e-14
	Std	7.0912e-5	2.3115e-5	1.71e-9	1.8932e-3	1.8025e-9	7.8221e+2	2.1269e-14	3.7130e-14
F5	1st	1.1133e-6	0	2.46e-9	1.1267e-8	6.6833e+2	0	0	0
	7th	0.0028	0	5.02e-9	1.7025e-7	8.4984e+2	0	0	0
	13th	0.0073	0	6.33e-9	4.9662e-7	9.5814e+2	0	0	0
	19th	0.0168	0	8.60e-9	9.4387e-7	1.0897e+3	3.6380e-12	0	0
	25th	0.0626	0	9.84e-9	1.0268e-5	1.9917e+3	7.8776e+2	0	0
	Mean	0.0123	0	6.58e-9	1.1383e-6	1.0049e+3	3.1557e+1	0	0
	Std	0.0146	0	2.17e-9	2.1828e-6	2.6176e+2	1.5754e+2	0	0
F6	1st	0	4.1410e-4	1.44e-9	6.0652e-11	9.2686e-5	4.4241e-3	7.6930e-2	1.4704e-9
	7th	4.3190e-9	2.4519e-1	3.81e-9	1.3616e-9	5.5107e-4	6.2559e-3	2.6544e+0	4.9131e-9
	13th	5.1631e-9	3.1617e-1	4.69e-9	3.0839e-9	4.0885e-2	1.5176e-2	2.9959e+0	8.9166e-9
	19th	9.1734e-9	5.0135e-1	5.67e-9	6.1451e-9	2.8477e+0	3.9973e+0	3.6026e+0	2.1826e-7
	25th	8.0479e-8	9.6508e-1	8.13e-9	1.6000e-6	7.6620e+0	8.9261e+2	4.4916e+0	6.2924e-6
	Mean	1.1987e-8	3.6579e-1	4.87e-9	6.8925e-8	1.2911e+0	9.5349e+1	2.9116e+0	7.1820e-7
	Std	1.9282e-8	2.2060e-1	1.66e-9	3.1904e-7	2.1051e+0	2.4543e+2	1.1309e+0	1.5252e-6
F7	1st	4.6700e-10	2.7195e+3	6.22e-10	4.8879e-9	3.1108e+3	5.6648e-2	2.8422e-14	0
	7th	1.48e-2	2.9486e+3	1.65e-9	2.2151e-2	3.3696e+3	1.3050e-1	7.3960e-3	3.9790e-13
	13th	1.97e-2	3.2102e+3	2.84e-9	4.4283e-2	3.5769e+3	2.0438e-1	9.8974e-3	7.3960e-3
	19th	2.71e-2	3.4067e+3	5.46e-9	6.6493e-2	3.9102e+3	3.6189e-1	1.7274e-2	1.2316e-2
	25th	3.69e-2	3.6533e+3	7.77e-9	1.2799e-1	4.6988e+3	7.4597e-1	5.4719e-2	3.2013e-2
	Mean	1.99e-2	3.1651e+3	3.31e-9	4.5189e-2	3.6983e+3	2.4563e-1	1.4193e-2	9.1745e-3
	Std	1.07e-2	2.7187e+2	2.02e-9	3.2611e-2	4.2224e+2	1.6324e-1	1.3503e-2	1.0578e-2
F8	1st	20.0000	2.0156e+1	2.00e+1	2.0000e+1	2.0083e+1	2.0099e+1	2.0126e+1	2.0000e+1
	7th	20.0000	2.0281e+1	2.00e+1	2.0000e+1	2.0137e+1	2.0211e+1	2.0278e+1	2.0000e+1
	13th	20.0000	2.0330e+1	2.00e+1	2.0000e+1	2.0167e+1	2.0272e+1	2.0362e+1	2.0000e+1
	19th	20.0000	2.0371e+1	2.00e+1	2.0000e+1	2.0268e+1	2.0306e+1	2.0403e+1	2.0000e+1
	25th	20.0000	2.0463e+1	2.00e+1	2.0000e+1	2.0474e+1	2.0423e+1	2.0465e+1	2.0001e+1
	Mean	20.0000	2.0326e+1	2.00e+1	2.0000e+1	2.0199e+1	2.0261e+1	2.0331e+1	2.0000e+1
	Std	5.3901e-8	7.4168e-2	3.89e-3	5.5382e-9	9.1104e-2	7.6041e-2	9.5430e-2	2.5382e-4

Table 1. (*Continued*)

Funcs.		L-SaDE	SACP-DE	LR-CMA-ES	DMS-L-PSO	CLPSO	SPSO-2007	HGO	L-HGO
	1st	0	0	1.52e-10	0	1.9899e+0	5.9697e+0	0	0
	7th	0	0	3.46e-10	0	3.9798e+0	9.9496e+0	0	0
	13th	0	0	6.14e-10	0	4.9748e+0	1.0945e+1	9.9496e-1	5.6843e-14
F9	19th	0	0	3.50e-9	0	5.9698e+0	1.4924e+1	9.9496e-1	9.9496e-1
	25th	0	0	9.95e-1	0	7.9597e+0	2.2884e+1	2.9849e+0	1.9899e+0
	Mean	0	0	2.39e-1	0	4.8554e+0	1.2594e+1	7.5617e-1	5.9698e-1
	Std	0	0	4.34e-1	0	1.5815e+0	4.3063e+0	7.7496e-1	7.0354e-1
	1st	1.9899	1.3243e+1	1.50e-10	1.9899e+0	7.9597e+0	3.9798e+0	2.0582e-3	0
	7th	3.9798	1.5268e+1	3.34e-10	2.9849e+0	1.1939e+1	7.9597e+0	9.9574e-1	9.9496e-1
	13th	4.9748	1.6779e+1	5.64e-10	3.9798e+0	1.5919e+1	1.3928e+1	1.9899e+0	9.9496e-1
F10	19th	5.9698	1.9010e+1	1.08e-9	3.9798e+0	2.1889e+1	1.7507e+1	2.9885e+0	2.9849e+0
	25th	9.9496	2.4500e+1	9.95e-1	4.9748e+0	4.0793e+1	2.3879e+1	5.0115e+0	3.9798e+0
	Mean	4.9685	1.7185e+1	7.96e-2	3.6217e+0	1.7471e+1	1.3444e+1	1.9615e+0	1.7511e+0
	Std	1.6918	3.0103e+0	2.75e-1	8.5590e-1	8.1546e+0	5.8035e+0	1.4263e+0	1.3250e+0
	1st	3.2352	5.7828e+0	5.27e-10	2.7590e+0	1.9484e-1	5.0886e-1	1.7635e-5	6.0004e-5
	7th	4.5129	6.7190e+0	3.48e-2	4.4767e+0	2.5144e+0	2.1478e+0	1.5000e-2	2.8510e-3
	13th	4.7649	7.0276e+0	6.34e-1	4.6570e+0	3.3422e+0	2.5552e+0	7.5498e-2	7.1042e-2
F11	19th	5.3823	7.3644e+0	1.64e+0	5.0608e+0	4.4543e+0	3.1957e+0	6.5254e-1	2.7160e-1
	25th	5.9546	8.2542e+0	3.19e+0	5.2394e+0	7.6678e+0	3.7362e+0	1.2664e+0	1.6480e+0
	Mean	4.8909	6.9982e+0	9.34e-1	4.6229e+0	3.4362e+0	2.4719e+0	3.7918e-1	2.9052e-1
	Std	0.6619	6.3794e-1	9.00e-1	5.8400e-1	1.7027e+0	9.0019e-1	4.4355e-1	4.7407e-1
	1st	1.4120e-10	1.7072e+3	1.08e-9	3.3481e-11	6.3715e-6	2.1138e+3	9.8920e+3	1.0158e-6
	7th	1.7250e-8	2.2332e+3	2.81e-9	3.0343e-10	1.4485e-1	7.6271e+3	1.5639e+4	1.1241e-3
	13th	8.1600e-8	2.5057e+3	3.89e-9	7.2993e-10	1.0003e+1	1.0146e+4	2.0941e+4	5.2651e-2
F12	19th	3.8878e-7	3.0837e+3	5.94e-9	1.4545e-5	1.8835e+1	1.7040e+4	2.5132e+4	1.3474e+3
	25th	3.3794e-6	4.1731e+3	7.12e+2	1.0003e+1	1.6936e+3	2.8260e+4	3.7099e+4	1.6936e+3
	Mean	4.5011e-7	2.6605e+3	2.93e+1	2.4007e+0	2.1071e+2	1.2214e+4	2.1366e+4	4.7516e+2
	Std	8.5062e-7	6.9549e+2	1.42e+2	4.3602e+0	4.9578e+2	6.6159e+3	7.5338e+3	7.2324e+2
	1st	0.1201	6.5789e-1	4.07e-1	2.5403e-1	1.1902e-1	3.7861e-1	3.9113e-1	2.4856e-1
	7th	0.1957	7.5483e-1	6.44e-1	3.2064e-1	4.6867e-1	6.6312e-1	5.4520e-1	3.7844e-1
	13th	0.2170	8.3828e-1	6.82e-1	3.6085e-1	5.9665e-1	7.4046e-1	5.7953e-1	4.0864e-1
F13	19th	0.2508	8.6047e-1	7.61e-1	4.0121e-1	8.1384e-1	9.9497e-1	7.2001e-1	4.7261e-1
	25th	0.3117	9.7245e-1	1.05e+0	4.7266e-1	1.2010e+0	1.8375e+0	9.9512e-1	7.2356e-1
	Mean	0.2202	8.1645e-1	6.96e-1	3.6865e-1	6.5480e-1	8.3801e-1	6.2690e-1	4.3591e-1
	Std	0.0411	8.0547e-2	1.50e-1	5.6411e-2	2.7688e-1	3.1776e-1	1.3407e-1	1.1888e-1
	1st	2.5765	3.1192e+0	2.08e+0	1.4838e+0	1.8246e+0	1.8263e+0	5.6454e-1	5.6628e-1
	7th	2.7576	3.4501e+0	2.75e+0	2.1160e+0	2.8009e+0	2.3163e+0	1.3294e+0	1.4408e+0
	13th	2.8923	3.5640e+0	3.00e+0	2.3522e+0	3.0708e+0	2.6537e+0	1.6906e+0	1.8625e+0
F14	19th	3.0258	3.6549e+0	3.28e+0	2.6055e+0	3.1780e+0	3.0165e+0	2.0670e+0	2.1318e+0
	25th	3.3373	3.7318e+0	3.51e+0	2.8906e+0	4.0127e+0	3.5063e+0	3.0726e+0	2.6985e+0
	Mean	2.9153	3.5240e+0	3.01e+0	2.3601e+0	3.0099e+0	2.6226e+0	1.7193e+0	1.7736e+0
	Std	0.2063	1.6791e-1	3.49e-1	3.3750e-1	4.7116e-1	4.5624e-1	6.6719e-1	5.3521e-1

After combined with local search, the performance of HGO is significantly improved especially for functions 6-9 and 12, and its success rate is raised from 0% to 100% for function 6, from 52% to 72% for function 7, from 40% to 52% for function 9, from 0% to 44% for function 12. Of course, the impact of the used local search schedule is not always positive; for example, its performance is deteriorated for functions 2, 4 and 14.

Table 2. Successful FEs and success performance of various algorithms

Funcs	Algorithms	1st	13th	25th	Mean	Std	Success Rate	Success Performance
F1	L-SaDE	10126	10126	10126	10126	0	100%	1.0126e+4
	SACP-DE	27347	28694	29944	2.8701e+4	6.7244e+2	100%	2.8701e+4
	LR-CMA-ES	1.44e+3	1.63e+3	1.71e+3	1.61e+3	6.14e+1	100%	1.61e+3
	DMS-L-PSO	1.1843e+4	1.1915e+4	1.1946e+4	1.1912e+4	2.5786 +1	100%	1.1912e+4
	CLPSO	19028	20678	22370	2.0702e+4	7.0589e+2	100%	2.0702e+4
	SPSO-2007	2541	2767	2995	2.7666e+3	1.1149e+2	100%	2.7666e+3
	HGO	6.5320e+3	7.2900e+3	8.4440e+3	7.3048e+3	4.4876e+2	100%	7.3048e+3
	L-HGO	3249	3249	3249	3249	0	100%	3249
F2	L-SaDE	10227	10241	10244	10237	7.2920	100%	1.0237e+4
	SACP-DE	94577	-	-	-	-	24%	4.0555e+5
	LR-CMA-ES	2.20e+3	2.35e+3	2.60e+3	2.38e+3	1.06e+2	100%	2.38e+3
	DMS-L-PSO	1.1843e+4	1.2046e+4	1.2340e+4	1.2052e+4	1.1488e+2	100%	1.2052e+4
	CLPSO	26966	28900	30721	2.8683e+4	1.0814e+3	100%	2.8683e+4
	SPSO-2007	6109	7723	9504	7.9303e+3	8.9394e+2	100%	7.9303e+3
	HGO	2.2121e+4	2.5232e+4	2.6734e+4	2.4951e+4	1.2441e+3	100%	2.4951e+4
	L-HGO	6.8340e+3	1.3652e+4	2.0448e+4	1.4835e+4	3.2041e+3	100%	1.4835e+4
F3	L-SaDE	28357	36644	-	-	-	64%	5.2306e+4
	SACP-DE	-	-	-	-	-	0%	-
	LR-CMA-ES	5.84e+3	6.51e+3	7.20e+3	6.50e+3	2.92e+2	100%	6.50e+3
	DMS-L-PSO	1.2165e+4	1.2519e+4	1.2649e+4	1.2480e+4	1.2440e+2	100%	1.2480e+4
	CLPSO	-	-	-	-	-	0%	-
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	-	-	-	-	-	0%	-
	L-HGO	-	-	-	-	-	0%	-
F4	L-SaDE	31040	39110	-	-	-	96%	4.5601e+4
	SACP-DE	-	-	-	-	-	0%	-
	LR-CMA-ES	2.52e+3	2.88e+3	3.22e+3	2.90e+3	1.68e+2	100%	2.90e+3
	DMS-L-PSO	-	-	-	-	-	0%	-
	CLPSO	28562	31977	35436	3.1827e+4	1.7093e+3	100%	3.1827e+4
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	2.7185e+4	2.9522e+4	3.1375e+4	2.9288e+4	1.2098e+3	100%	2.9288e+4
	L-HGO	2.6605e+4	2.9317e+4	3.1809e+4	2.9202e+4	1.4748e+3	100%	2.9202e+4
F5	L-SaDE	-	-	-	-	-	0%	-
	SACP-DE	47405	49409	52414	4.9656e+4	1.4919e+6	100%	4.9656e+4
	LR-CMA-ES	5.36e+3	5.83e+3	6.72e+3	5.85e+3	2.89e+2	100%	5.85e+3
	DMS-L-PSO	7.7125e+4	8.9185e+4	-	-	-	80%	1.1336e+5
	CLPSO	-	-	-	-	-	0%	-
	SPSO-2007	3859	4875	-	-	-	88%	1.0912e+4
	HGO	7.3970e+3	9.2620e+3	1.0783e+4	9.3535e+3	7.6755e+2	100%	9.3535e+3
	L-HGO	7.9050e+3	1.0651e+4	1.2340e+4	1.0312e+4	1.1795e+3	100%	1.0312e+4
F6	L-SaDE	31546	52756	63382	48777	10240	100%	4.8777e+4
	SACP-DE	83171	-	-	-	-	8%	1.0918e+6
	LR-CMA-ES	5.67e+3	8.55e+3	2.26e+4	1.08e+4	5.00e+3	100%	1.08e+4
	DMS-L-PSO	2.5912e+4	5.2022e+4	7.8003e+4	5.4677e+4	1.2942e+4	100%	5.4677e+4
	CLPSO	63447	-	-	-	-	48%	1.5553e+5
	SPSO-2007	86051	-	-	-	-	36%	2.4532e+5
	HGO	-	-	-	-	-	0%	-
	L-HGO	4.4816e+4	4.8016e+4	6.0727e+4	4.8268e+4	4.0065e+3	100%	4.8268e+4
F7	L-SaDE	10257	-	-	-	-	24%	1.7197e+5
	SACP-DE	-	-	-	-	-	0%	-
	LR-CMA-ES	1.49e+3	5.83e+3	1.33e+4	4.67e+3	2.83e+3	100%	4.67e+3
	DMS-L-PSO	4.9529e+4	-	-	-	-	16%	5.8672e+5
	CLPSO	-	-	-	-	-	0%	-
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	1.5700e+4	4.9381e+4	-	-	-	52%	6.5136e+4
	L-HGO	6.8340e+3	2.7288e+4	-	-	-	72%	3.1849e+4

Table 2. (*Continued*)

Funcs	Algorithms	1st	13th	25th	Mean	Std	Success Rate	Success Performance
F9	L-SaDE	14540	17217	20103	17048	1340.9	100%	1.7048e+4
	SACP-DE	46532	52241	58120	5.2126e+4	3.0352e+3	100%	5.2126e+4
	LR-CMA-ES	2.33e+4	7.85e+4	-	-	-	76%	7.57e+4
	DMS-L-PSO	2.4651e+4	3.6919e+4	4.9663e+4	3.4612e+4	8.8073e+3	100%	3.4612e+4
	CLPSO	-	-	-	-	-	0%	-
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	3.8625e+4	-	-	-	-	40%	1.1130e+5
	L-HGO	4.3002e+4	6.6480e+4	-	-	-	52%	1.0691e+5
F10	L-SaDE	-	-	-	-	-	0%	-
	SACP-DE	-	-	-	-	-	0%	-
	LR-CMA-ES	2.68e+4	5.15e+4	-	-	-	92%	6.50e+4
	DMS-L-PSO	-	-	-	-	-	0%	-
	CLPSO	-	-	-	-	-	0%	-
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	6.8971e+4	-	-	-	-	12%	6.0441e+5
	L-HGO	5.3240e+4	-	-	-	-	16%	4.2706e+5
F11	L-SaDE	-	-	-	-	-	0%	-
	SACP-DE	-	-	-	-	-	0%	-
	LR-CMA-ES	3.05e+4	-	-	-	-	24%	2.63e+5
	DMS-L-PSO	-	-	-	-	-	0%	-
	CLPSO	-	-	-	-	-	0%	-
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	2.3123e+4	-	-	-	-	24%	1.2489e+5
	L-HGO	2.6401e+4	-	-	-	-	36%	9.4848e+4
F12	L-SaDE	10302	31493	52733	31933	12789	100%	3.1933e+4
	SACP-DE	-	-	-	-	-	0%	-
	LR-CMA-ES	2.37e+3	3.10e+4	-	-	-	88%	3.27e+4
	DMS-L-PSO	1.2482e+4	2.5687e+4	-	-	-	76%	5.4443e+4
	CLPSO	46864	-	-	-	-	16%	4.2168e+5
	SPSO-2007	-	-	-	-	-	0%	-
	HGO	-	-	-	-	-	0%	-
	L-HGO	1.0243e+4	-	-	-	-	44%	7.7500e+4

4 Conclusions

The HGO is a novel swarm intelligence algorithm by simulating human behaviors, especially human searching/foraging behaviors. In order to improve the local search ability, the canonical HGO is combined with the Quasi-Newton method. The studies on the comparisons between L-HGO and other algorithms are also conducted on several functions provided by CEC05. The simulation results document that the proposed algorithm is a competitive and promising candidate of search algorithms. In the future research, we will further conduct in-depth study on human social behaviors, especially human searching/foraging behaviors, for solving optimization problems.

Acknowledgments. This work was supported in part by the Fundamental Research Funds for the Central Universities under Contracts (SWJTU09CX026) and Experimental Education and Technology Project of Southwest Jiaotong University (No. 2010A03).

References

1. Kordon, A.K.: Swarm intelligence: The benefits of swarms. In: Applying Computational Intelligence: How to Create Value, pp. 145–174. Springer, Heidelberg (2010)
2. Timmis, J., Andrews, P., Hart, E.: On artificial immune systems and swarm intelligence. Swarm Intell. 4, 247–273 (2010)
3. Krause, J., Ruxton, G.D., Krause, S.: Swarm intelligence in animals and humans. Trends in Ecology & Evolution 25, 28–34 (2010)
4. Goldstone, R.L., Roberts, M.E., Gureckis, T.M.: Emergent processes in group behavior. Current Directions in Psychological Science 17, 10–15 (2008)
5. Dai, C.H., Chen, W.R., Song, Y.H., et al.: Seeker optimization algorithm: A novel stochastic search algorithm for global numerical optimization. J. Syst. Eng. Electro. 21, 300–311 (2010)
6. Cohen, J.D., McClure, S.M., Yu, A.J.: Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. Philos. Trans. R. Soc. Lond., Ser. B: Biol. Sci. 362, 933–942 (2007)
7. Dai, C., Zhu, Y., Chen, W.: Seeker optimization algorithm. In: Wang, Y., Cheung, Y., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 167–176. Springer, Heidelberg (2007)
8. Dai, C., Chen, W., Cheng, Z., et al.: Seeker optimization algorithm for global optimization: a case study on optimal modelling of proton exchange membrane fuel cell (PEMFC). Int. J. Electr. Power Energ. Syst. 33, 369–376 (2011)
9. Dai, C., Chen, W., Zhu, Y., et al.: Reactive power dispatch considering voltage stability with seeker optimization algorithm. Electr. Power Syst. Res. 79, 1462–1471 (2009)
10. Dai, C., Chen, W., Zhu, Y., et al.: Seeker optimization algorithm for optimal reactive power dispatch. IEEE Trans. Power Syst. 24, 1218–1231 (2009)
11. Sivasubramani, S., Swarup, K.S.: Hybrid SOA–SQP algorithm for dynamic economic dispatch with valve-point effects. Energy 35(12), 5031–5036 (2010)
12. Shaw, B., Mukherjee, V., Ghoshal, S.P.: Seeker optimization algorithm: application to the solution of economic load dispatch problems. IET Gener. Transm. Distrib. 5, 81–91 (2011)
13. Krishnanand, K.R., Rout, P.K., Panigrahi, B.K., Mohapatra, A.: Solution to non-convex electric power dispatch problem using seeker optimization algorithm. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) SEMCCO 2010. LNCS, vol. 6466, pp. 537–544. Springer, Heidelberg (2010)
14. Zhao, Z., Li, Y., Yu, J., et al.: Optimal assembly tolerance design based on Fuzzy information entropy and seeker optimization algorithm. In: 3rd International Conference on Advanced Computer Theory and Engineering, vol. 5, pp. 610–613 (2010)
15. Dai, C., Chen, W., Zhu, Y.: Seeker optimization algorithm for digital IIR filter design. IEEE Trans. Ind. Electron. 57, 1710–1718 (2010)
16. Dai, C., Chen, W., Zhu, Y., et al.: Seeker optimization algorithm for tuning the structure and parameters of neural networks. Neurocomputing 74, 876–883 (2011)
17. Dai, C., Chen, W., Ma, L., et al.: Human group optimizer for global numerical optimization. Int. J. Artif. Intell. Tools (2010) (in press)
18. Clerc, M.: Back to random topology (2007), `http://clerc.maurice.free.fr/pso/`
19. Standard PSO 2007 (SPSO-07) on the Particle Swarm Central, Programs section, `http://www.particleswarm.info/` (Login time in 2008)
20. Liang, J.J., Suganthan, P.N.: Dynamic multi-swarm particle swarm optimizer with local search. In: Proc. 2005 IEEE Congress on Evol. Comput., vol. 2, pp. 522–528 (2005)

21. Suganthan, P.N., Hansen, N., Liang, J.J., et al.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore (May 2005)
22. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proc. 2005 IEEE Congress on Evol. Comput (CEC 2005), Edinburgh, Scotland, vol. 2, pp. 1785–1791 (2005)
23. Brest, J., Greiner, S., Bošković, B., et al.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. 10, 646–657 (2006)
24. Auger, A., Kern, S., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proc. 2005 IEEE Congress on Evol. Comput (CEC 2005), Edinburgh, Scotland, vol. 2, pp. 1769–1776 (2005)
25. Liang, J.J., Qin, A.K., Suganthan, P.N., et al.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. 10, 67–82 (2006)

Average-Inertia Weighted Cat Swarm Optimization

Maysam Orouskhani[1,*], Mohammad Mansouri[2], and Mohammad Teshnehlab[3]

[1] Msc Student, Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
orouskhani@ce.sharif.edu
[2] Intelligent System Laboratory (ISLAB), faculty of Electrical Engineering, Control department, K.N. Toosi University of Technology, Tehran, Iran
[3] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

Abstract. For improving the convergence of Cat Swarm Optimization (CSO), we propose a new algorithm of CSO namely, Average-Inertia Weighted CSO (AICSO). For achieving this, we added a new parameter to the position update equation as an inertia weight and used a new form of the velocity update equation in the tracing mode of algorithm. Experimental results using Griewank, Rastrigin and Ackley functions demonstrate that the proposed algorithm has much better convergence than pure CSO.

Keywords: Cat Swarm Optimization, Average-Inertia Weighted Cat Swarm Optimization, Swarm Intelligence.

1 Introduction

Optimization and functions minimization are very important problems. So there are many algorithms to solve these problems.Some of these optimization algorithms were developed based on swarm intelligence bysimulating the intelligent behavior of animals, like Ant Colony Optimization (ACO) [1-6] which imitates the behavior of ants, Particle Swarm Optimization (PSO) [2] which imitates thebehavior of birds, Bee Colony Optimization (BCO)[3] which imitates the behavior of bees and the recent finding, Cat SwarmOptimization (CSO) [4] which imitates the behavior of cats.

In order to solve the optimization problems, CSO models the behavior of cats into two sub-models namely seeking mode and tracing mode.

In the cases of functions optimization, CSO is one of the best algorithms to find the best global solution. In comparison with other heuristic algorithms such as PSO and PSO with weighting factor [7], CSO usually achieves better result. But sometimes in some cases pure CSO takes a long time to find an acceptable solution. So it affects on performance and convergence of the algorithm.Therefore high speed processor is needed for getting reasonable result.

In this article, our aim is to introduce a new version of CSO in order to improve the performance and achieve better convergence in less iteration. First we add a new

*Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 321–328, 2011.
© Springer-Verlag Berlin Heidelberg 2011

parameter to the position equation as an inertia weight that will be chosen randomly (ICSO). Then by making a new form of the velocity equation composing two terms, we introduce Average-Inertia Weighted CSO (AICSO).

Experimental results indicate that the proposed algorithm rather than pureCSO can improve performance on finding the best global solution and achieves better accuracy level of convergence.

2 Cat Swarm Optimization

Chu et al. [4] divided CSO algorithm into two sub-models based on two of the major behavioral traits of cats.These are termed"seeking mode" and "tracing mode".In CSO, we first decide how many cats we would like to usein the iteration, then we apply the cats into CSO to solve the problems. Every cat has its own position composed of Ddimensions, velocities for each dimension, a fitness value, which represents the accommodation of the cat to the fitness function, and a flag to identify whether the cat is in seeking mode or tracing mode. The final solution would be the best position of one of the cats. The CSO keeps the best solution until it reaches the end of the iterations [5].

A. Seeking Mode

This sub model is used to model the cat during a period of resting but being alert-looking around its environment for its next move.

Seeking mode has four essential factors, which are designed as follows: seeking memory pool (SMP), seeking range of the selected dimension (SRD), counts of dimension to change (CDC) and self position consideration (SPC).

Seeking mode is described below.

Step 1: Make j copies of the present position of cat_k , where j = SMP. If the value of SPC is true, let j = (SMP-1), then retain the present position as one of the candidates.

Step 2:For each copy, according to CDC, randomly plus or minus SRD percent the present values and replace the old ones.

Step 3: Calculate the fitness values (FS) of all candidate points.

Step 4: If all FS are not exactly equal, calculate the selecting probability of each candidate point by equation (1), otherwise set all the selecting probability of each candidate point be 1.

Step 5: Randomly pick the point to move to from the candidate points, and replace the position of cat_k.

$$P = \frac{|SSE_i - SSE_{max}|}{SSE_{max} - SSE_{min}}, \qquad 0 < i < j \tag{1}$$

If the goal of the fitness function is to find the minimum solution,$FS_b = FS_{max}$, otherwise$FS_b = FS_{min}$

B. Tracing Mode

Tracing mode is the sub-model for modeling the case of the cat in tracing targets. The action of tracing mode can be described as follows:

Step 1: Update the velocities for every dimension ($v_{k,d}$) according to equation (2).

Step 2: Check if the velocities are in the range of maximum velocity. In case the new velocity is over-range, it is set equal to the limit.

$$v_{k,d} = v_{k,d} + r_1 c_1 \left(x_{best,d} - x_{k,d}\right) \tag{2}$$

Step 3: Update the position of cat_k according to (3)

$$\mathbf{x_{k,d}} = \mathbf{x_{k,d}} + \mathbf{v_{k,d}} \tag{3}$$

C. Core Description of CSO

In order to combine these two modes into the algorithm, we define a mixture ratio (MR) which dictates the joining of seeking mode with tracing mode. The MR decides how many cats will be moved by seeking mode process. For example,if the population size is 50 and the MR is equal to 0.7, there should be50*0.7=35 cats moveto seeking mode and 15 cats move to tracing mode in this iteration. So the process of CSO is summarized below:

First we create N cats and initialize the position, velocities and the flag of every cat. (*) According to the fitness function, evaluate the fitness value of the each cat and keep the best cat into memory. In next step, according to cat's flag, apply cat to the seeking mode or tracing mode process. After finishing the related process, re-pick the number of cats and set them into seeking mode or tracing mode according to MR. In the final step check the termination condition, if satisfied, terminate the program, otherwise go to (*).

3 AICSO Algorithm

In the pure CSO, we should have a condition on the velocity equation in order to control the velocities of the cats for every dimension and check whether the velocities are in the range of maximum or not.

For modifying this part, we use a parameter as an inertia weight to handle this problem.Here we choose the value of inertia weight (w) randomly and experimental results indicate that it is better to choose w in the range of [0.4,0.9].So we select the largest value for w in the first iteration (w = 0.9) and then it will be reduced to 0.4 in thenext iterations.CSO with inertia weight can converge under certain conditions even without using v_{max}. For w>1, velocities increase over time, causing cats to diverge eventually beyond the boundaries of the search space.Forw<1,velocities decrease over time,eventually reaching 0,resulting in convergence behavior.So the new position update equation can be written as

$$v_{k,d} = wv_{k,d} + r_1 c_1 \left(x_{best,d} - x_{k,d}\right) \tag{4}$$

Where $c_1$is acceleration coefficient and usually is equal to 2.05 and $r_1$is a random value uniformly generated in the range of [0,1] and w is inertia weight(ICSO).

Next step, we use a new form of the position update equation composingtwo terms. In the first term, the average information of current and previous position and in the second, the average of current and previous velocity information will be used(AICSO). So new position equation is described below

$$x_{i+1} = \frac{(x_{i+1} + x_i)}{2} + \frac{(v_{i+1} + v_i)}{2} \tag{5}$$

4 Experimental Results

In this paper, simulation experiments are conducted in GCC compiler on an Intel Core 2 Duo at 2.53GHz with 4G real memory.Our test is about finding the global minima of three test functions: Rastrigrin, Griewank and Ackley. In table1 we determine the parameter values of the algorithm and the limitations range of three test functions are mentioned in table2.

Table 1. Parameters settings(1)

Parameter	Value or range
SMP	5
SRD	20%
CDC	80%
MR	2%
c_1	2.05
r_1	[0,1]
w	[0.4,0.9]

Table 2. Parameters settings(2)

Function	Limitation Range	Object
Rastrigrin	[2.56,5.12]	Minimize
Griewank	[300,600]	Minimize
Ackley	[-32.768,32.768]	Minimize

Rastrigin's function is based on the function of DeJong with the addition of cosine modulation in order to produce frequent local minima. Thus, the test function is highly multimodal. Function has the following definition

$$f(x) = \sum_{d=1}^{M} [x_d{}^2 - 10\cos(2\pi x_d)^2 + 10] \tag{6}$$

Where -5.12<x<5.12, its global minimum is equal to 0.

As shown in Fig.1, AICSO in the 900th iteration achieves the best solution and its fitness value is so close to the answer of Rastrigin's function. So in comparison with pure CSO and ICSO, AICSO has so much better solution and takes less iteration to converge.

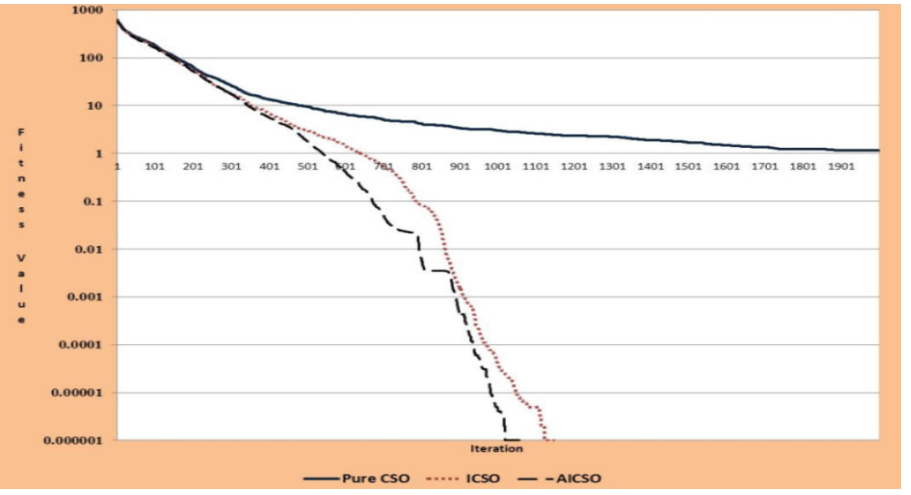


Fig. 1. The experimental result of Rastrigin function

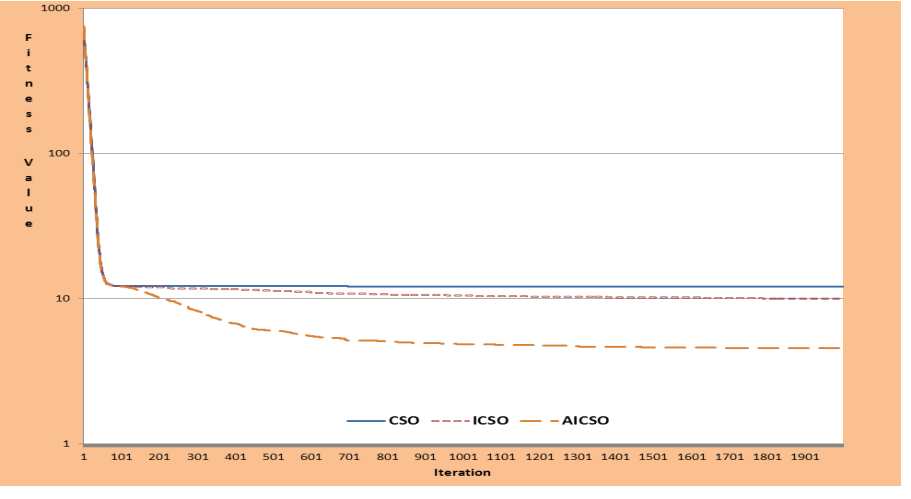


Fig. 2. The experimental result of Griewank function

Griewangk's function has many minima regularly distributed. Where -600<x<600, its global minimum is equal to 0. Function has the following definition

$$f(x) = \frac{1}{4000}\sum_{d=1}^{M} {x_d}^2 - \prod_{d=1}^{M} cos\left(\frac{x_d}{\sqrt{d}}\right) + 1 \tag{7}$$

As demonstrated in Fig.2, Pure CSO and ICSO have the same fitness value approximately but ICSO can find the better solution and its final solution is 4.57.

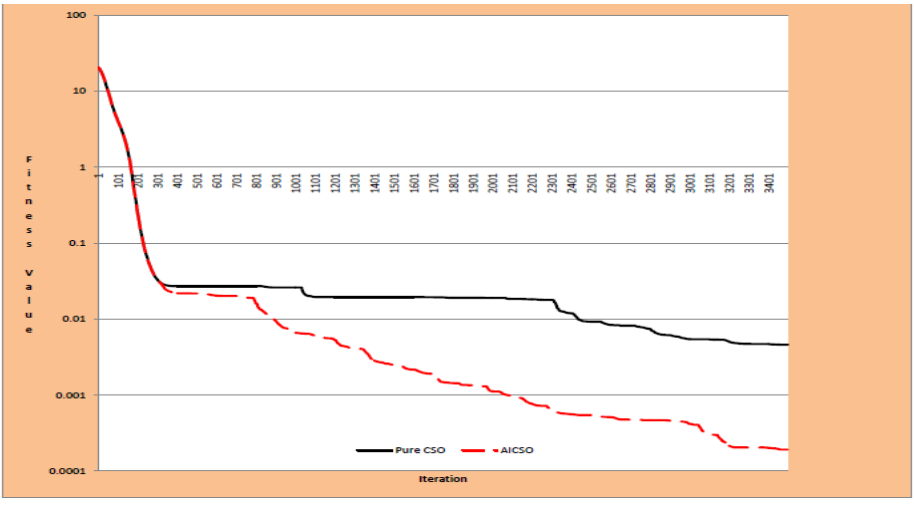


Fig. 3. The experimental result of Ackley function

Ackley's function is a widely used multimodal test function and Where32.68<x<32.68, its global minimum is equal to 0. It has the following definition

$$f(x) = -20.\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) - 20 + \exp(1) \quad (8)$$

As shown in Fig.3 AICSO rather than Pure CSO can find the better solution and the best fitness value is 0.000192 in the last iteration.

With comparisons of the average fitness value, numbers of iteration and the best global solution for CSO, ICSO and AICSO which are mentioned in Table3, table4 and table5,results indicate that AICSO has better performance and usually takes less iteration to converge rather than pure CSO and ICSO. For example, global results of Griewank function are shown in table4. The best fitness values of CSO, ICSO and AICSO are 12.08, 10.01, 4.57 respectively. So it is clear that AICSO has much better result and its solution is more acceptable than CSO and ICSO.

Table 3. Global Results of Rastrig in Function

Algorithm	Average Solution	Iteration	Best Solution
CSO	*26.64208*	*2000*	*1.161127*
ICSO	*22.45198*	*1100*	*0.000000*
AICSO	*21.718749*	*950*	*0.000000*

Table 4. Global Results of Griewank Function

Algorithm	Average Solution	Iteration	Best Solution
CSO	*16.34888*	*2000*	*12.081966*
ICSO	*1485353*	*2000*	*10.017866*
AICSO	*9.937433*	*2000*	*4.572061*

Table 5. Global Results of Ackley Function

Algorithm	Average Solution	Iteration	Best Solution
CSO	*0.38096*	*3500*	*0.004589*
AICSO	*0.370672*	*3500*	*0.000192*

5 Conclusions

Function minimization is a very important problem in the optimization theory. Cat Swarm Optimization is one of the useful algorithms to solve these problems. But pure CSO in some cases takes a long time to converge and in some problems cannot find the best global solution correctly. So we modified the tracing mode of the algorithm. To do this, we added an inertia weightto the velocity update equation (ICSO) and then we changed the position update equation to a new form using average of current and previous position/ velocity information (AICSO). Experimental results for three benchmarks indicated that the proposed algorithm in comparison with pure CSO and ICSO improves the performance on finding the best global solution and achieves the better accuracy level of convergence in the less iteration.

References

1. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to thetraveling salesman problem. IEEE Trans. on EvolutionaryComputation 26(1), 53–66 (1997)
2. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Sixth International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
3. Karaboga, D.: An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report-TR06,ErciyesUniversity, Engineering Faculty, Computer Engineering Department (2005)
4. Chu, S.C., Tsai, P.W., Pan, J.S.: Cat swarm optimization. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS (LNAI), vol. 4099, pp. 854–858. Springer, Heidelberg (2006)

5. Santosa, B., Ningrum, M.: Cat Swarm Optimization for Clustering. In: International Conferenceof Soft Computing and Pattern Recognition, pp. 54–59 (2009)
6. Chu, S.C., Roddick, J.F., Pan, J.S.: Ant colony system with communication strategies. Information Sciences 167, 63–76 (2004)
7. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Congress on Evolutionary Computation, pp. 1945–1950 (1999)
8. Molga, M., Smutnicki, C.: Test functions for optimization needs, 3 kwietnia (2005)

Standby Redundancy Optimization with Type-2 Fuzzy Lifetimes

Yanju Chen and Ying Liu

College of Mathematics and Computer Science, Hebei University
Baoding 071002, Hebei, China
yanjuchen@hbu.edu.cn

Abstract. Under the given system weight constraint, we consider the problem of maximizing the system lifetime and minimizing the system cost. The lifetimes of components in the system are characterized by type-2 fuzzy variables. The numbers of redundant elements of each components are the decision variables. We use the reduction methods to reduce the type-2 fuzzy lifetimes. Then, we propose a goal programming model for this system. We suggest an approximation approach (AA) to the reliability and design an AA-based particle swarm optimization (PSO) algorithm to solve the fuzzy model.

Keywords: Standby redundancy optimization, Type-2 fuzzy variables, Generalized credibility, Approximation approach, Particle swarm optimization.

1 Introduction

In various kinds of systems, reliability engineering is especially important. The primary goal of reliability engineering is to improve the reliability of a system. The redundancy allocation is a direct way of enhancing the binary-state system reliability. There exist two basic ways to provide some components redundancy: parallel redundancy and standby redundancy. In the first way, all redundant elements are operating while in the second way, one of the redundant elements begins to work only when the active element has failed.

Based on the assumption that the elements' lifetimes are random variables, many lots of researches have been done. Coit and Smith [1] have studied redundancy allocation to maximize a lower percentile of the system time-to-failure distribution. Zhao and Liu [2] have considered both parallel redundant systems and standby redundant systems, three types of system performance are introduced. Reliability has become a greater concern in recent years, Kuo and Wan [3] provides a broad overview of recent research on reliability optimization problems and their solution methodologies. To enhance the reliability of the systems in which the lifetimes are imprecise or vague, some researchers used the fuzzy set theory to study fuzzy reliability optimization problems [4,5,6]. Mahapatra and Roy [7] discussed a fuzzy multi-objective optimization method for a multi-objective system reliability problem. With different optimization criteria, Zhao

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 329–337, 2011.
© Springer-Verlag Berlin Heidelberg 2011

and Liu [8] built three kinds of standby redundancy optimization models for a standby redundancy system with fuzzy lifetimes.

Since the membership function is difficult to be determined, Zadeh [9] introduced the concept of a type-2 fuzzy set as an extension of an ordinary fuzzy set. A type-2 fuzzy set is characterized by a fuzzy membership function. Liu and Liu [10] presented fuzzy possibility theory. In fuzzy possibility theory, a variable-based approach is adopted to deal with type-2 fuzziness. To characterize the properties of type-2 fuzzy variables in some aspects, Chen and Wang [11] presented a scalar representative value operator for type-2 fuzzy variable. To defuzzify type-2 fuzzy variables, Qin et al. [12,13] gave the mean reduction methods and the critical value reduction methods for the type-2 fuzzy variable.

This paper models the redundancy optimization based on fuzzy possibility theory, the lifetimes of components are characterized by type-2 fuzzy variables with known secondary possibility distributions.

The paper is organized as follows. Section 2 recalls some basic concepts and results in fuzzy possibility theory. In Section 3, we formulate a multi-objective redundancy optimization model with type-2 fuzzy lifetimes. Then we use the reduction methods to reduce the type-2 fuzzy lifetimes and propose a goal programming model. Section 4 employs the AA to evaluate the system reliability and designs an approximation-based PSO algorithm to solve the proposed goal programming model. One numerical example is provided in Section 5. Finally, Section 6 summarizes the main work in this paper.

2 Preliminaries

Let ξ be a general fuzzy variable with the distribution μ. The generalized credibility measure [13], denoted by $\tilde{\mathrm{Cr}}$, of the event $\{\xi \geq r\}$ is defined by

$$\tilde{\mathrm{Cr}}\{\xi \geq r\} = \frac{1}{2}\left(\sup_{x\in\Re}\mu(x) + \sup_{x\geq r}\mu(x) - \sup_{x<r}\mu(x)\right),\ r \in \Re.$$

The general fuzzy variables $\xi_1, \xi_2, \ldots, \xi_n$ are said to be mutually independent [13] if

$$\tilde{\mathrm{Cr}}\{\xi_i \in B_i, i = 1, 2, \ldots, n\} = \min_{1\leq i\leq n} \tilde{\mathrm{Cr}}\{\xi_i \in B_i\}$$

for any subsets B_i, $i = 1, 2, \ldots, n$ of $\Re$.

To reduce the type-2 fuzziness, one way is to give a reasonable representation for the secondary possibility distribution [10]. For this purpose, Qin et al. [13] have used the expectations [14] of secondary possibility distribution as the representing values and given the mean reduction methods for the type-2 fuzzy variable.

With E reduction method, the reduction η of type-2 normal fuzzy variable $\tilde{\eta} = \tilde{n}(\mu, \sigma^2; \theta_l, \theta_r)$ has the following distribution [13]

$$\mu_\eta(x) = \begin{cases} \frac{(4+\theta_r-\theta_l)\exp(-\frac{(x-\mu)^2}{2\sigma^2})}{4}, & if\ x \leq \mu - \sigma\sqrt{2\ln 2}\ or\ x \geq \mu + \sigma\sqrt{2\ln 2} \\ \frac{(4-\theta_r+\theta_l)\exp(-\frac{(x-\mu)^2}{2\sigma^2})+\theta_r-\theta_l}{4}, & if\ \mu - \sigma\sqrt{2\ln 2} < x < \mu + \sigma\sqrt{2\ln 2}. \end{cases}$$

With E reduction method, the reduction ξ of type-2 triangular fuzzy variable $\tilde{\xi} = (\tilde{r}_1, \tilde{r}_2, \tilde{r}_3; \theta_l, \theta_r)$ has the following distribution [13]

$$\mu_\xi(x) = \begin{cases} \frac{(4+\theta_r-\theta_l)(x-r_1)}{4(r_2-r_1)}, & if\ x \in [r_1, \frac{r_1+r_2}{2}] \\ \frac{(4-\theta_r+\theta_l)x+(\theta_r-\theta_l)r_2-4r_1}{4(r_2-r_1)}, & if\ x \in (\frac{r_1+r_2}{2}, r_2] \\ \frac{(-4+\theta_r-\theta_l)x+4r_3-(\theta_r-\theta_l)r_2}{4(r_3-r_2)}, & if\ x \in (r_2, \frac{r_2+r_3}{2}] \\ \frac{(4+\theta_r-\theta_l)(r_3-x)}{4(r_3-r_2)}, & if\ x \in (\frac{r_2+r_3}{2}, r_3]. \end{cases}$$

3 Fuzzy Standby Redundancy Optimization Problem

This paper considers a standby redundant system containing five components which is showed in Fig. 1. Assume that the ith component consists of x_i redundant elements, $i = 1, 2, \ldots, 5$, respectively. Firstly, we give the following assumptions: 1) The system, components, and elements, at any time, are binary-state (good or failed). 2) There is no element (or system) repair or preventive maintenance. 3) The failures of the elements are mutually-independent. 4) The switching device of the standby system is perfect. 5) The elements of the same type have independent and identically distributed lifetimes, the lifetimes of each components are independent.

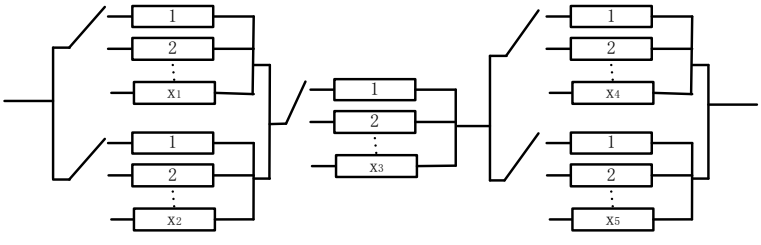


Fig. 1. A standby redundant system

Sometimes, we can only obtain limited information about the lifetimes such as the distributions. In this section, we assume that we can only obtain the type-2 distributions of the lifetimes, i.e., the lifetimes are characterized by type-2 fuzzy variables. We use type-2 fuzzy vector $\tilde{\xi} = (\tilde{\xi}_{1,1}, \ldots, \tilde{\xi}_{1,x_1}, \ldots, \tilde{\xi}_{5,1}, \ldots, \tilde{\xi}_{5,x_5})$ where $\tilde{\xi}_{i,j}$ is the type-2 fuzzy lifetime of the jth element in component i for all $i = 1, 2, \ldots, n, j = 1, 2, \ldots, x_i$, to characterize the lifetimes of the components. The system lifetime at allocation $x = (x_1, \ldots, x_5)$ can be expressed as

$$\begin{aligned} & T(x, \tilde{\xi}) \\ = & \max\{(\tilde{\xi}_{1,1} + \cdots + \tilde{\xi}_{1,x_1}) \wedge (\tilde{\xi}_{3,1} + \cdots + \tilde{\xi}_{3,x_3}) \wedge (\tilde{\xi}_{4,1} + \cdots + \tilde{\xi}_{4,x_4}), \\ & (\tilde{\xi}_{1,1} + \cdots + \tilde{\xi}_{1,x_1}) \wedge (\tilde{\xi}_{3,1} + \cdots + \tilde{\xi}_{3,x_3}) \wedge (\tilde{\xi}_{5,1} + \cdots + \tilde{\xi}_{5,x_5}), \\ & (\tilde{\xi}_{2,1} + \cdots + \tilde{\xi}_{2,x_2}) \wedge (\tilde{\xi}_{3,1} + \cdots + \tilde{\xi}_{3,x_3}) \wedge (\tilde{\xi}_{4,1} + \cdots + \tilde{\xi}_{4,x_4}), \\ & (\tilde{\xi}_{2,1} + \cdots + \tilde{\xi}_{2,x_2}) \wedge (\tilde{\xi}_{3,1} + \cdots + \tilde{\xi}_{3,x_3}) \wedge (\tilde{\xi}_{5,1} + \cdots + \tilde{\xi}_{5,x_5})\}. \end{aligned} \quad (1)$$

Our problem is to find the optimal redundancy allocations to this system so as to maximize the system lifetime and minimize the system cost. Based on those two different objectives, we can formulate the standby redundancy optimization model as

$$\begin{cases} \max & T(x,\tilde{\xi}) \\ \min & \sum_{i=1}^{5} c_i x_i + \sum_{i=1}^{5} c_i \exp(h_i x_i) \\ \text{subject to} & \sum_{i=1}^{5} w_i x_i + \sum_{i=1}^{5} w_i \exp(m_i x_i) \leq w_0 \\ & x \geq \mathbf{1}, \text{ integer vector} \end{cases} \tag{2}$$

where $\mathbf{1} = (1,1,1,1,1)$, x_i is the decision variable which denotes the number of elements in the ith component, w_0 denotes the maximal weight the system can burden, w_i denotes the weight of element in the ith component, c_i denotes the cost of element in the ith component, $w_i \exp(m_i x_i)$ and $c_i \exp(h_i x_i)$ denote the additional weight and cost due to the interconnection between elements in the ith component, respectively, h_i, m_i represent the corresponding parameters.

The problem (2) is not well-defined because the meaning of the first objective is not clear. In this sense, solving such a programming is meaningless. In order to obtain a meaningful mathematical model, we can use the mean reduction methods to reduce the type-2 fuzzy lifetimes. We expect to maximize the system lifetime and minimize the system cost. Let T_0 be the threshold duration of the system lifetime, and $\tilde{\mathrm{Cr}}\{T(x,\xi) \geq T_0\}$ the system reliability. If we regard system reliability as the first priority, then we obtain the following goal programming model

$$\begin{cases} \text{lexmin} & \{d_1^-, d_2^+\} \\ \text{subject to} & \tilde{\mathrm{Cr}}\{T(x,\xi) \geq T_0\} + d_1^- - d_1^+ = \alpha_0 \\ & \sum_{i=1}^{5} c_i x_i + \sum_{i=1}^{5} c_i \exp(h_i x_i) + d_2^- - d_2^+ = c_0 \\ & \sum_{i=1}^{5} w_i x_i + \sum_{i=1}^{5} w_i \exp(m_i x_i) \leq w_0 \\ & x \geq \mathbf{1}, \text{ integer vector} \\ & d_i^-, d_i^+ \geq 0, \ i = 1,2 \end{cases} \tag{3}$$

where $\xi = (\xi_{1,1}, \ldots, \xi_{1,x_1}, \ldots, \xi_{5,1}, \ldots, \xi_{5,x_5})$ and $\xi_{i,j}$ is the reduced fuzzy variable of $\tilde{\xi}_{i,j}$ according to the mean reduction methods, $\tilde{\mathrm{Cr}}$ is the generalized credibility measure, α_0 is the given level the system reliability should achieve, c_0 is the given level of the system cost should achieve.

4 Solution Method

To solve the fuzzy goal programming model (3), it is required to evaluate the generalized credibility function $\tilde{\mathrm{Cr}}\{T(x,\xi) \geq T_0\}$. For any feasible decision x, we evaluate $\tilde{\mathrm{Cr}}\{T(x,\xi) \geq T_0\}$ by using the AA [15]. Then we suggest an approximation-based PSO algorithm to solve the model (3).

4.1 An Approximation Scheme

For simplicity, we rewrite $\xi = (\xi_1, \xi_2, \ldots, \xi_m)$ where $m = x_1 + \cdots + x_5$. Suppose that ξ is an essentially bounded fuzzy vector, and the possibility distributions ν_i

of $\xi_i, i = 1, 2, \ldots, m$, are continuous on $\Re$. Let μ be the possibility distribution of ξ, and defined as

$$\mu(u_1, u_2, \ldots, u_m) = \min_{1 \le i \le m} \nu_i(u_i) \tag{4}$$

for every $(u_1, u_2, \ldots, u_m) \in \Re^m$.

Denote $\Xi = \prod_{i=1}^{m} [a_i, b_i]$ as the support of ξ with $[a_i, b_i]$ the supports of ξ_i, $i = 1, 2, \ldots, m$, respectively.

For any given integer n, the discrete fuzzy vector $\zeta_n = (\zeta_{n,1}, \zeta_{n,2}, \ldots, \zeta_{n,m})$ is defined as

$$\zeta_n = h_n(\xi) = (h_{n,1}(\xi_1), h_{n,2}(\xi_2), \ldots, h_{n,m}(\xi_m)) \tag{5}$$

where the fuzzy variables $\zeta_{n,i} = h_{n,i}(\xi_i),\ i = 1, 2, \ldots, m$, with

$$h_{n,i}(u_i) = \sup \left\{ \tfrac{k_i}{n} \mid k_i \in Z, \tfrac{k_i}{n} \le u_i \right\} \tag{6}$$

for $u_i \in [a_i, b_i]$, and Z the set of all integers.

Moreover, for each $i, 1 \le i \le m$, by the definition of $\zeta_{n,i}$, as ξ_i takes its values in $[a_i, b_i]$, the fuzzy variable $\zeta_{n,i}$ takes its values in the set $\{\frac{k_i}{n} \mid k_i = [na_i], [na_i] + 1, \ldots, K_i\}$, where $[r]$ is the maximal integer such that $[r] \le r$, and $K_i = nb_i - 1$ or $[nb_i]$ according as nb_i is an integer or not an integer. Therefore, the possibility distribution $\nu_{n,i}$ of $\zeta_{n,i}$ is

$$\nu_{n,i}\left(\tfrac{k_i}{n}\right) = \mathrm{Pos}\left\{\gamma \mid \tfrac{k_i}{n} \le \xi_i(\gamma) < \tfrac{k_i+1}{n}\right\} \tag{7}$$

for $k_i = [na_i], [na_i] + 1, \ldots, K_i$. As a consequence, the possibility distribution μ_n of $\zeta_n = (\zeta_{n,1}, \ldots, \zeta_{n,m})$ is

$$\mu_n\left(\tfrac{k_1}{n}, \tfrac{k_2}{n}, \ldots, \tfrac{k_m}{n}\right) = \min_{1 \le i \le m} \nu_{n,i}\left(\tfrac{k_i}{n}\right) \tag{8}$$

for $i = 1, 2, \ldots, m$, and $k_i = [na_i], [na_i] + 1, \ldots, K_i$.

The sequence $\{\zeta_n\}$, which converges uniformly to the continuous fuzzy vector ξ [15], is referred to as the *discretization* of the continuous fuzzy vector ξ.

4.2 Computing the Generalized Credibility

Let $\tilde{\mathrm{Cr}}\{T(x, \xi) \ge T_0\}$ is denoted as $\tilde{\mathrm{Cr}}\{g(\xi) \ge T_0\}$. We now discuss the computation of $\tilde{\mathrm{Cr}}\{g(\xi) \ge T_0\}$ according to the following two cases.

Case 1. Suppose $\xi = (\xi_1, \xi_2, \ldots, \xi_m)$ has the possibility distribution

$$\xi \sim \begin{pmatrix} \hat{\xi}_1 & \hat{\xi}_2 & \ldots & \hat{\xi}_K \\ \nu_1 & \nu_2 & \ldots & \nu_K \end{pmatrix} \tag{9}$$

with $\nu_k = \mathrm{Pos}\{\xi = \hat{\xi}_k\} > 0$, $\hat{\xi}_k = (\hat{\xi}_{k,1}, \hat{\xi}_{k,2}, \ldots, \hat{\xi}_{k,m}) \in \Re^m$, and $\max_{k=1}^{K} \nu_k = \sup_{x \in \Xi} \mu(x)$. The generalized credibility

$$\begin{aligned} &\tilde{\mathrm{Cr}}\{g(\xi) \ge T_0\} \\ &= \tfrac{1}{2}(\sup_{x \in \Xi} \mu(x) + \max\{\nu_k \mid g(\hat{\xi}_k) \ge T_0\} - \max\{\nu_k \mid g(\hat{\xi}_k) < T_0\}) \end{aligned}$$

where $\Xi = \left\{\hat{\xi}_1, \hat{\xi}_2, \ldots, \hat{\xi}_K\right\} \subset \Re^m$.

Case 2. Suppose the support Ξ of ξ is continuous. We utilize the following approach to compute $L = \tilde{\mathrm{Cr}}\{g(\xi) \geq T_0\}$.

Let $\{\zeta_n\}$ be the *discretization* of the fuzzy vector ξ. For each fixed n, the fuzzy vector ζ_n takes on K values $\hat{\zeta}_n^k = (\hat{\zeta}_{n,1}^k, \hat{\zeta}_{n,2}^k, \ldots, \hat{\zeta}_{n,m}^k), k = 1, 2, \ldots, K$, with $K = K_1 K_2 \cdots K_m$. Write $\nu_k = \nu_{n,1}(\hat{\zeta}_{n,1}^k) \wedge \nu_{n,2}(\hat{\zeta}_{n,2}^k) \wedge \cdots \wedge \nu_{n,m}(\hat{\zeta}_{n,m}^k)$ for $k = 1, 2, \ldots, K$, where $\nu_{n,i}$ defined by Eq. (7) are the possibility distributions of the fuzzy variables $\zeta_{n,i}, i = 1, 2, \ldots, m$, respectively. Then $\tilde{\mathrm{Cr}}\{g(\zeta_n) \geq T_0\}$ can be computed by the formula

$$\frac{1}{2}\left(\sup_{x\in\Xi}\mu(x) + \max\{\nu_k \mid g(\hat{\zeta}_n^k) \geq T_0\} - \max\{\nu_k \mid g(\hat{\zeta}_n^k) < T_0\}\right). \tag{10}$$

Liu [15] shows that $\tilde{\mathrm{Cr}}\{g(\zeta_n) \geq T_0\}$ converges to $\tilde{\mathrm{Cr}}\{g(\xi) \geq T_0\}$ as $n \to \infty$, which implies that $\tilde{\mathrm{Cr}}\{g(\xi) \geq T_0\}$ can be estimated by the formula (10) provided n is sufficiently large. The process to estimate the generalized credibility $\tilde{\mathrm{Cr}}\{g(\xi) \geq T_0\}$ (i.e., AA) is summarized as

Step 1. Generate K points $\hat{\zeta}_n^k = (\hat{\zeta}_{n,1}^k, \hat{\zeta}_{n,2}^k, \ldots, \hat{\zeta}_{n,m}^k)$, $k = 1, 2, \ldots, K$, which form the support of ζ_n.
Step 2. Calculate $g(\hat{\zeta}_n^k)$ for $k = 1, 2, \ldots, K$.
Step 3. Set $\nu_k = \nu_{n,1}(\hat{\zeta}_{n,1}^k) \wedge \nu_{n,2}(\hat{\zeta}_{n,2}^k) \wedge \cdots \wedge \nu_{n,m}(\hat{\zeta}_{n,m}^k)$ for $k = 1, 2, \ldots, K$.
Step 4. Return L via the estimation formula (10).

4.3 Hybrid Particle Swarm Optimization

PSO algorithm, originally developed in [16], is based on *pop_size* particles, each of which indicates a possible solution of the problem space. Each particle has its own best position (*pbest*) which represents the personal smallest objective value so far at time t. The global best particle (*gbest*) represents the best particle found so far at time t in the colony. The new velocity of the ith particle is updated by the following formula

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 \left(P_i(t) - X_i(t)\right) + c_2 r_2 \left(P_g(t) - X_i(t)\right) \tag{11}$$

for $i = 1, 2, \ldots, pop_size$, where ω is called the inertia coefficient, c_1 and c_2 are the learning rates and usually $c_1 = c_2 = 2$, and r_1, r_2 are two independent random numbers generated from the unit interval [0,1]. Let $X = (x_1, \ldots, x_5)$ be denoted as a particle to represent the decision vector. Since model (3) is an integer programming, the new position of the ith particle is renewed by

$$X_i(t+1) = X_i(t) + IV_i(t+1) \tag{12}$$

where

$$IV_i(t+1) = \begin{cases} 1, & if\ V_i(t+1) > 0 \\ -1, & if\ V_i(t+1) < 0 \\ 0, & if\ V_i(t+1) = 0. \end{cases}$$

The solution process of PSO combined AA is summarized as follows.

Step 1. Initialize *pop_size* feasible particles with random positions and velocities, then compute their objective values by AA.

Step 2. Set *pbest* of each particle and its objective value equal to its current position and objective value, and set *gbest* and its objective value equal to the position and objective value of the best initial particle;

Step 3. Renew the velocity and position of each particle according to formulas (11) and (12), respectively.

Step 4. Calculate the objective values for all particles by AA.

Step 5. For each particle, compare the current objective value with that of its *pbest*. If the current objective value is smaller than that of *pbest*, then renew *pbest* and its objective value with the current position and objective value.

Step 6. Find the best particle of the current swarm with the smallest objective value. If the objective value is smaller than that of *gbest*, then renew *gbest* and its objective value with the position and objective value of the current best particle.

Step 7. Repeat the third to six steps for a given number of cycles.

Step 8. Return the *gbest* and its objective value as the optimal solution and the optimal value.

5 A Numerical Example

We consider a standby redundant system containing five components which is showed in Fig. 1. Assume that the ith component consists of x_i redundant elements, $i = 1, 2, \ldots, 5$, respectively. For the ith component, the lifetime of each element is characterized by type-2 fuzzy variable $\tilde{\xi}_i$, which is shown in Table 1, $\tilde{\xi}_i,\ i = 1, 2, \ldots, 5$ are mutually independent type-2 fuzzy variables [10]. Let ξ_i $(i = 1, 2, \ldots, 5)$ be the reduced fuzzy variable of $\tilde{\xi}_i$ according to the E reduction method, and the support of ξ_3 be $[0, 15]$. Furthermore, we suppose that the threshold duration of the system lifetime is $T_0 = 5$, the prescribed confidence level of the system reliability is $\alpha_0 = 0.9$, the given cost level of the system should achieve is $c_0 = 150$, the maximal weight the system can burden is $w_0 = 220$, the parameters $h_i = m_i = 0.3, i = 1, \ldots, 5$. The other parameters are also shown in Table 1. Then we build the following goal programming model

$$\begin{cases} \text{lexmin} & \{d_1^-, d_2^+\} \\ \text{subject to} & \tilde{\text{Cr}}\{T(x,\xi) \geq 5\} + d_1^- - d_1^+ = 0.9 \\ & \sum_{i=1}^5 c_i x_i + \sum_{i=1}^5 c_i \exp(0.3x_i) + d_2^- - d_2^+ = 150 \\ & \sum_{i=1}^5 w_i x_i + \sum_{i=1}^5 w_i \exp(0.3x_i) \leq 220 \\ & x \geq \mathbf{1},\ \text{integer vector} \\ & d_i^-, d_i^+ \geq 0,\ i = 1, 2. \end{cases} \tag{13}$$

For simplicity, we assume that $\theta_l = \theta_l^i, \theta_r = \theta_r^i, i = 1, 2, \ldots, 5$. For some values of (θ_l, θ_r), the AA-based PSO algorithm (1000 generations in PSO) shows the optimal solutions of model (13) which are provided in Table 1.

Table 1. Parameters and solutions

i	$\tilde{\xi}_i$	c_i	w_i	(θ_l, θ_r)	optimal solution x	d_1^-	d_2^+
1	$(\tilde{3}, \tilde{6}, \tilde{8}; \theta_l^1, \theta_r^1)$	10	5	$(0.6, 0.5)$	$(3, 1, 3, 1, 2)$	0	46.7643
2	$(\tilde{4}, \tilde{6}, \tilde{7}; \theta_l^2, \theta_r^2)$	12	6	$(0.4, 0.8)$	$(2, 1, 3, 1, 2)$	0	46.7643
3	$\tilde{n}(5, 2; \theta_l^3, \theta_r^3)$	13	8	$(0.8, 0.5)$	$(2, 1, 3, 1, 2)$	0	46.7643
4	$(\tilde{3}, \tilde{4}, \tilde{8}; \theta_l^4, \theta_r^4)$	9	5	$(0.4, 0.7)$	$(4, 1, 4, 2, 1)$	0	104.4587
5	$(\tilde{4}, \tilde{6}, \tilde{7}; \theta_l^5, \theta_r^5)$	10	6	$(0, 0)$	$(2, 1, 2, 1, 2)$	0.0098	25.4770

6 Conclusions

The paper's innovation contents include the following three aspects:

(a) Based on fuzzy possibility theory, we model the standby redundancy optimization problem and use the mean reduction methods to reduce the type-2 fuzzy lifetimes with known secondary possibility distributions. Then based on generalized credibility measure, we give a goal programming model;
(b) When the lifetimes of components are characterized by mutually independent type-2 fuzzy variables, we employ the AA to estimate the system reliability. Then we design an approximation-based PSO algorithm to solve the proposed goal programming model;
(c) For the proposed model, we provide one numerical example to illustrate the modeling idea and the efficiency of the proposed algorithm.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.60974134), the Natural Science Foundation and the Education Department of Hebei Province (No.A2011201007, No.F2010000318, No.2010109), and the Scientific Research and Development Program of Baoding (No.10ZR002).

References

1. Coit, D.W., Smith, A.E.: Redundancy Allocation to Maximize a Lower Percentile of the System Time-to-Failure Distribution. IEEE Transactions on Reliability 47, 79–87 (1998)
2. Zhao, R., Liu, B.: Stochastic Programming Models for General Redundancy Optimization Problems. IEEE Transactions on Reliability 52, 181–192 (2003)
3. Kuo, W., Wan, R.: Recent Advances in Optimal Reliability Allocation. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 37(2), 143–156 (2007)
4. Pedrycz, W., Gomide, F.: Fuzzy Systems Engineering: Toward Human-Centric Computing. J. Wiley, Hoboken (2007)
5. Zadeh, L.A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)
6. Li, B., Zhu, M., Xu, K.: A Practical Engineering Method for Fuzzy Reliability Analysis of Mechanical Structures. Reliability Engineering & System Safety 67, 311–315 (2000)

7. Mahapatra, G.S., Roy, T.K.: Fuzzy Multi-Objective Mathematical Programming on Reliability Optimization Model. Applied Mathematics and Computation 174, 643–659 (2006)
8. Zhao, R., Liu, B.: Standby Redundancy Optimization Problems with Fuzzy Lifetimes. Computers & Industrial Engineering 49, 318–338 (2005)
9. Zadeh, L.A.: Concept of a Linguistic Variable and Its Application to Approximate Reasoning I. Information Sciences 8, 199–249 (1975)
10. Liu, Z.Q., Liu, Y.-K.: Type-2 Fuzzy Variables and Their Arithmetic. Soft Computing 14(7), 729–747 (2009)
11. Chen, Y., Wang, X.: The Possibilistic Representative Value of Type-2 Fuzzy Variable and Its Properties. Journal of Uncertain Systems 4(3), 229–240 (2010)
12. Qin, R., Liu, Y.K., Liu, Z.Q.: Methods of Critical Value Reduction for Type-2 Fuzzy Variables and Their Applications. Journal of Computational and Applied Mathematics 235(5), 1454–1481 (2011)
13. Qin, R., Liu, Y.K., Liu, Z.Q.: Modeling Fuzzy Data Envelopment Analysis by Parametric Programming Method. Expert Systems with Applications 38(7), 8648–8663 (2011)
14. Liu, B., Liu, Y.-K.: Expected Value of Fuzzy Variable and Fuzzy Expected Value Models. IEEE Transactions on Fuzzy Systems 10(4), 445–450 (2002)
15. Liu, Y.-K.: Convergent Results about the Use of Fuzzy Simulation in Fuzzy Optimization Problems. IEEE Transactions on Fuzzy Systems 14(2), 295–304 (2006)
16. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)

Oriented Search Algorithm for Function Optimization

Xuexia Zhang and Weirong Chen

School of Electrical and Engineering, Southwest Jiaotong University, 610031 Chengdu, China
zxxswjtu@gmail.com, wrchen@swjtu.edu.cn

Abstract. A population-based algorithm, oriented search algorithm (OSA), is proposed to optimize functions in this paper. In OSA, the search-individual imitates human random search behavior, and the search-object simulates an intelligent agent that can transmit oriented information to search-individuals. OSA is tested on thirteen complex benchmark functions. The results are compared with those of particle swarm optimization with inertia weight (PSO-w), particle swarm optimization with constriction factor (PSO-cf) and comprehensive learning particle swarm optimizer (CLPSO). The results show that OSA is superior in convergence efficiency, search precision, convergence property and has the strong ability to escape from the local sub-optima.

Keywords: swarm intelligence, oriented search algorithm, human random search behavior, function optimization.

1 Introduction

As a novel computational intelligence technology, swarm intelligence [1-4] has become more and more researchers' focus. Swarm intelligence [3-5] means that simple individuals show the character of intelligent behavior through co-operation between individuals. Swarm intelligence behavior is the self-organizing act which is not controlled by central controller, meaning that the collective behavior emerges from simple individuals interacting locally with one another and with their environment [6-7]. Recently, the algorithms based on swarm intelligence mainly include ant colony algorithm [8-9], particle swarm optimization [10-18] etc. Shi Y. and Eberhart R. [19] proposed particle swarm optimization with inertia weight (PSO-w) which can balance between local optima and global optima. Nevertheless, the inertial weight linearly decreased with run time increasing affects the global search capability. Clerc M. and Kennedy J. [20] presented particle swarm optimization with constriction factor (PSO-cf), which improves the search ability of particle swarm optimization. Liang J.J. and Qin A.K. [21] introduced comprehensive learning particle swarm optimizer (CLPSO), which advances the ability of escaping from local optima. However, it has shortcomings in unimodal functions optimization.

The authors have proposed a novel algorithm based on population, oriented search algorithm (OSA), to optimize reactive power dispatch in power system [22-23]. In order to verify the performance of OSA, function optimization based on OSA is presented in this paper. Trough testing thirteen benchmark functions based on OSA,

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 338–346, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the results identify that OSA has high convergence efficiency, search precision, convergence property and the strong ability to escape from the local sub-optima.

2 Oriented Search Algorithm

In OSA [22-23], the search-individual simulates human random search behavior, and the search-object (i.e., the optimal solution of the objective functions) works like an intelligent agent that can transmit oriented information to search-individuals. The oriented information transmitted by the current-search-object makes the search-individual search towards better direction. At the same time, the search-object continuously adjusts its position so that search-individuals can receive the variable oriented information to search the optimal object as soon as possible. In a word, OSA simulates the search behavior of human searching intelligent agent to model the communication between search-object and search-individuals.

2.1 Search-Individual

Search-individual simulates human random search behavior. The search-individual moves discretely in multidimensional space. During the search process, each search-individual takes exploration walks and every walk includes the search step and the search direction. A number of steps determine the search period. The initial position of the search-individual can be defined as follows:

$$x_{0ji} = X_{\min i} + (X_{\max i} - X_{\min i}) * random(0,1) \ . \tag{1}$$

Where $random(0,1)$ denotes a random value between 0 and 1. $X_{\min i}$ and $X_{\max i}$ denote the bounds of search space. i=1, …, n, n denotes the dimension of search space, i denotes the ith dimension. j=1, …, m, m denotes the population size, j denotes the jth search-individual.

2.2 Search-Object

In each walk, all the search-individuals obtain one current-search-object, the current optimal solution of the objective function. The current-search-object adjusts its position ($x_{tglobal}$) adaptively for transmitting the oriented information to search-individuals. At the same time, the oriented information acts as the oriented-neighbor-space of the next walk for search-individuals.

Search-object sends oriented information to search-individuals in order to reduce the search range of the individual blindly searching. According to the variable oriented information, search-individuals continuously generate stochastic search steps and search directions to update their positions.

2.3 Oriented-Neighbor-Space

Oriented-neighbor-space is a random neighbor-space built by the position of the current-search-object. In each walk, the current-search-object adjusts its position

adaptively, at the same time to send the information of its position, called oriented information, to search-individuals. In same walk, each search-individual receives different oriented information. Therefore, oriented-neighbor-space is formed for each search-individual. Moreover, in different walk, oriented information is changeable. As a result, oriented-neighbor-space is variable as well. For the whole exploration process, the range of the oriented-neighbor-space is reduced gradually with search-individuals approaching to the search-object, and becomes one certain position until search-individuals achieve the search-object. It means that search-individuals, receiving the oriented information transmitted by the current-search-object in every current walk, are getting concentrative and inclined to the search-object step by step, and find the search-object until the certain position information of the search-object was accepted. Oriented-neighbor-space is defined as follows:

$$x_{tglobal} * (1 + w * randn(0,1)) . \tag{2}$$

Where $randn(0,1)$ denotes a random number with normal distributed between 0 and 1. w denotes a variable parameter to regulate the random changing trend of oriented-neighbor-space. The minimal w is zero, which denotes search-individuals achieve the optimal position of search-object when optimization process is ended. w is defined as follows:

$$w = w_{\max} - t * (w_{\max} - w_{\min}) / G . \tag{3}$$

Where t is the current generation number, G is the maximal generation number.

2.4 Search-Neighbor-Space, Updating Strategy of Search Direction and Search Step

The search-neighbor-space of every search-individual is related closely with the oriented-neighbor-space. According to the oriented-neighbor-space, search-individuals set search-neighbor-space, and generate stochastic search steps and search directions. In search-neighbor-space, different search-individual generates different search direction which points to its own current-search-object. Also, the search step is stochastic distance between the current position of the search-individual and the position of its own current-search-object. The updating strategy of search direction and search step is described as follows:

$$\Delta x_{tji} = (x_{tglobal} * (1 + w * randn(0,1)) - x_{tji}) * random(0,1) . \tag{4}$$

2.5 Evaluation and Decision

Throughout each exploration, every search-individual achieves a solution of the objective function which is evaluated by evaluation function. Then, the result of the evaluation for each solution of the objective function marks the quality of the solution. If the quality of the solution is better, the current position of the search-individual is better. Furthermore, the decision function is applied to determine whether updating the current search walk or not. If the current position of search-individual is better than that of last walk, it will be updated. After each walk, the

current-search-object is updated through selecting the best quality of the solution among the solutions of the objective function. The search process continues until the termination criterion is met.

Explore new position of the current search-individual $x_{(t+1)ji}$:

$$x_{(t+1)ji} = x_{tji} + \Delta x_{tji} . \tag{5}$$

Evaluate the quality of the objective function solution:

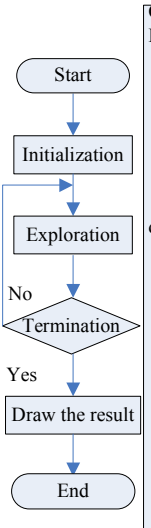
$$f_{tj} = f(x_{tji}) . \tag{6}$$

Decide whether generating search behavior:

$$x_{tji} = \begin{cases} x_{(t+1)ji}, (if: f_{(t+1)j} \le f_{tj}) \\ x_{tji}, \quad (if: f_{(t+1)j} > f_{tj}) \end{cases} . \tag{7}$$

2.6 OSA Architecture

The structure of OSA mainly includes three parts: initialization, exploration and termination. The algorithm architecture is presented in Fig.1 as a flow chart.



Oriented Search {
Initialization{
initialize $X_{\min i}$, $X_{\max i}$, $n,m,G;$
initialize the position of search individuals x_{0ji};
generate an initial solution of the objective function for each search individual $f(x_{0ji})$;
generate an initial optimal solution of the objective function for population $\min(f(x_{0ji}))$;
initialize w}//end initialization
do{ // **exploration**
generate the strategy of updating Δx_{tji};
explore new position of the current search individual $x_{(t+1)ji}$;
evaluate the quality of the objective function solution f_{tj};
decide whether generating search behavior
$x_{tji} = \begin{cases} x_{(t+1)ji}, (if: f_{(t+1)j} \le f_{tj}) \\ x_{tji}, \quad (if: f_{(t+1)j} > f_{tj}) \end{cases}$
update the current position of the objective function optimal solution $x_{tglobal}$;
} while $((f_{\min} \le f_{opt}) \| (g \ge G))$ //**termination**
output the result}
// end Oriented Search

Fig. 1. Algorithm architecture

3 Simulation and Analysis

OSA is tested on thirteen complex benchmark functions and the results are compared with those obtained by particle swarm optimization with inertia weight (PSO-w), particle swarm optimization with constriction factor (PSO-cf) and comprehensive

learning particle swarm optimizer (CLPSO). The parameters settings are list as follows. The parameters of PSO-w are set as: learning rate is $c_1 = c_2 = 2$, inertia weight w is linearly decreased from 0.9 to 0.4 with implementation time increasing, the maximum velocity $v_{\max}$ is set at 20% of the dynamic range of the variable on each dimension; the parameters of PSO-cf are set as: learning rate is $c_1 = c_2 = 2.01$, constriction factor is $\chi = 0.729844$; the parameters of CLPSO are set as: $c = 1.49445$, inertia weight w is linearly decreased from 0.9 to 0.4 with implementation time increasing; the parameter of OSA is set as: inertia weight w is linearly decreased from 0.9 to 0 with implementation time increasing. Experiments are implemented in MATLAB 7.0 with Lenovo PC of Pentium(R) 4 CPU 2.94GHz, 512MB of RAM.

Table 1 lists thirteen functions, where n is the dimension of variable, S is the bound of variable and $f_{\min}$ is optimal solution of objective function in theory. Table 2, 3 and table 4 list the optimal results of unimodal functions, multimodal functions with

Table 1. Thirteen benchmark functions

Function	n	S	$f$min
$f1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	30	$[-5.12,5.12]^n$	0
$f_2(\vec{x}) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	$[-100,100]^n$	0
$f_3(\vec{x}) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	$[-30,30]^n$	0
$f_4(\vec{x}) = \sum_{i=1}^{n} i x_i^4 + rand[0,1)$	30	$[-1.28,1.28]^n$	0
$f_5(\vec{x}) = -\sum_{i=1}^{n} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	30	$[-500,500]^n$	−12569.5
$f_6(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	$[-5.12,5.12]^n$	0
$f_7(\vec{x}) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	$[-32,32]^n$	0
$f_8(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600,600]^n$	0
$f_9(\vec{x}) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	30	$[-50,50]^n$	0
$f_{10}(\vec{x}) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50,50]^n$	0
$f_{11}(\vec{x}) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2]$	6	$[0,1]^n$	−3.32
$f_{12}(\vec{x}) = -\sum_{i=1}^{7} [(\vec{x} - a_i)(\vec{x} - a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	−10.4029
$f_{13}(\vec{x}) = -\sum_{i=1}^{10} [(\vec{x} - a_i)(\vec{x} - a_i)^T + c_i]^{-1}$	4	$[0,10]^n$	−10.5364

many local minima and multimodal functions with only a few local minima, respectively. The same population size is 100 and a total of 50 runs are made in all simulations. The maximal generation (G), the best value (Best), the mean value (Mean), the standard deviation (Std. Dev.) and CPU implementation time (Time (Second)) are illustrated, respectively.

Table 2 shows that OSA has better ability of searching Best, Mean and Std. Dev. for unimodal functions than other algorithms but the Best of function f_3.

Table 3 shows that OSA has better capability of exploring Best, Mean and Std. Dev. for multimodal functions with many local minima than other algorithms but the Best of functions f_9 and f_{10}.

Table 4 shows that OSA has better facility of finding the Best of function f_{11} and the Mean of function f_{12} and f_{13} for multimodal functions with only a few local minima than other algorithms but the Std. Dev..

From the index of CPU implementation time, we can see that PSO-w has slightly better than other algorithms.

From the integrated performance of OSA, OSA is better in the convergence efficiency, search precision, convergence property and strong ability to escape from the local sub-optima.

Table 2. The performance comparison of algorithms for unimodal functions

Function	Algorithm	Best	Mean	Std. Dev.	Time(second)
f_1 (G=1500)	PSO-w	1.7864e-015	1.6568e-013	4.5931e-013	**18.180**
	PSO-cf	4.5023e-045	2.2848e-041	4.5442e-041	19.817
	CLPSO	3.2147e-013	2.7272e-012	1.684e-012	24.450
	OSA	**9.1166e-124**	**1.5372e-120**	**2.6619e-120**	27.088
f_2 (G =5000)	PSO-w	0.011794	0.07022	0.046572	**63.410**
	PSO-cf	1.4824e-016	7.1273e-013	2.1876e-012	73.237
	CLPSO	0.00068793	0.002049	0.0012533	83.920
	OSA	**4.7762e-101**	**4.2001e-092**	**2.8823e-091**	101.960
f_3 (G =20000)	PSO-w	0.010498	1821.6	12727	**251.490**
	PSO-cf	**1.8661e-012**	7323.6	24636	271.720
	CLPSO	0.16786	36.256	31.223	349.100
	OSA	23.299	**25.786**	**1.2625**	502.920
f_4 (G =3000)	PSO-w	0.0029905	0.0062767	0.0021719	**42.960**
	PSO-cf	0.00098644	0.0024501	0.0013784	51.558
	CLPSO	0.0010339	0.0029838	0.00097204	57.380
	OSA	**3.7417e-005**	**0.00013706**	**6.2126e-005**	78.754

Table 3. The performance comparison of algorithms for multimodal functions with many local minima

Function	Algorithm	Best	Mean	Std. Dev.	Time(second)
f_5 (G =9000)	PSO-w	-10495	-9363.3	445.34	**126.650**
	PSO-cf	-10398	-9026.1	656.9	150.250
	CLPSO	-12569	-12271	**177.84**	158.870
	OSA	**-7528**	**-6386.4**	621.12	241.220
f_6 (G =5000)	PSO-w	7.9597	21.036	8.0054	**66.950**
	PSO-cf	26.864	61.714	18.406	71.912
	CLPSO	2.8525e-010	1.3387e-009	8.5715e-010	87.740
	OSA	**0**	**0**	**0**	112.68
f_7 (G =1500)	PSO-w	1.3927e-007	1.6594e-006	2.6584e-006	**20.970**
	PSO-cf	2.6645e-015	0.55865	0.72955	22.528
	CLPSO	3.3065e-006	6.812e-006	1.9421e-006	27.090
	OSA	**-8.8818e-016**	**7.4607e-016**	**1.7886e-015**	39.001
f_8 (G =2000)	PSO-w	0	0.015917	0.021873	**28.520**
	PSO-cf	0	0.01112	0.012484	30.884
	CLPSO	1.6431e-014	0.00029587	0.001464	36.660
	OSA	**0**	**0**	**0**	54.004
f_9 (G =1500)	PSO-w	8.8539e-015	2.2084	5.523	**28.990**
	PSO-cf	**1.5705e-032**	16.62	18.099	31.940
	CLPSO	8.7972e-012	**4.802e-011**	**3.9566e-011**	35.160
	OSA	0.014073	0.058148	0.032146	39.950
f_{10} (G =1500)	PSO-w	8.2301e-007	572.42	357.41	36.978
	PSO-cf	**1.3498e-032**	239.83	240.31	**33.646**
	CLPSO	1.1773e-010	**6.4202e-010**	**4.4627e-010**	38.618
	OSA	0.5063	0.9541	0.23547	39.897

Table 4. The performance comparison of algorithms for multimodal functions with only a few local minima

Function	Algorithm	Best	Mean	Std. Dev.	Time(second)
f_{11} (G =200)	PSO-w	-3.322	-3.2561	0.065727	2.831
	PSO-cf	-3.322	**-3.2767**	**0.058388**	**2.755**
	CLPSO	-3.322	-3.2744	0.058837	3.537
	OSA	**-3.321**	-3.2582	0.059905	4.848
f_{12} (G =100)	PSO-w	-4.6069	-2.1373	**0.83394**	**1.250**
	PSO-cf	**-10.403**	-6.4735	3.5574	1.410
	CLPSO	-10.339	-9.4026	1.1175	1.910
	OSA	-10.398	**-9.7249**	0.98188	2.814
f_{13} (G =100)	PSO-w	-6.6324	-2.2004	**1.0073**	**1.350**
	PSO-cf	**-10.536**	-8.1124	3.4669	1.506
	CLPSO	-10.457	-9.4725	1.2532	2.010
	OSA	-10.372	**-9.6482**	1.4202	3.024

4 Conclusion

Oriented search algorithm simulates the search behavior of human searching intelligent agent, which is different from other swarm intelligent algorithms. In OSA, there is one parameter w to be adjusted except population size, maximal generation. And it is easier to implement than other algorithms. Through testing on thirteen benchmark functions, the results demonstrate that the OSA has advantages in the convergence efficiency, search precision, convergence property and the strong ability to escape from the local sub-optima.

References

1. Beni, G., Wang, J.: Swarm Intelligence. In: Proc. of the Seventh Annual Meeting of the Robotics Society of Japan, pp. 425–428 (1989)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: from Natural to Artificial Systems. Oxford University Press, New York (1999)
3. Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for Optimization from Social Insect Behavior. Nature 406, 39–42 (2002)
5. Michael, G., Hinchey, R.S., Chris, R.: Swarms and Swarm Intelligence. Computer 40, 111–113 (2007)
6. Kristina, L., Aram, G.: A General Methodology for Mathematical Analysis of Multi-agent Systems. USC Information Sciences Technical Report ISI-TR-529 (2001)
7. Bonabeau, E., Meyer, C.: Swarm Intelligence: A Whole New Way to Think About Business. Harvard Business Review, 106–114 (2001)
8. Colorni, A., Dorigo, M., Maniezze, V.: Distributed Optimization by Ant Colonies. In: Proc. of the 1st European Conf. Artificial Life, Pans, France, pp. 134–142. Elsevier, Amsterdam (1991)

9. Dorigo, M., Blum, C.: Ant Colony Optimization Theory: A Survey. Theoretical Computer Science 344, 243–278 (2005)
10. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
11. Eberhart, R., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proc. of the Sixth International Symposium on Micromachine and Human Science, pp. 39–43 (1995)
12. Zhao, B., Guo, C.X., Cao, Y.J.: A Multi-agent Based Particle Swarm Optimization Approach for Reactive Power Dispatch. IEEE Trans. Power Syst. 20, 1070–1078 (2005)
13. Esmin, A.A.A., Lambert-Torres, G., Zambroni de Souza, A.C.: A Hybrid Particle Swarm Optimization Applied to Loss Power Minimization. IEEE Trans. Power Syst. 20, 859–866 (2005)
14. John, G., Vlachogiannis, K.Y.L.: A Comparative Study on Particle Swarm Optimization for Optimal Steady-state Performance of Power Systems. IEEE Trans. Power Syst. 21, 1718–1728 (2006)
15. Coelho, L., dos, S., Herrera, B.M.: Fuzzy Identification Based on A Chaotic Particle Swarm Optimization Approach Applied to A Nonlinear Yo-yo Motion System. IEEE Trans. Ind. Electron. 54, 3234–3245 (2007)
16. Del, V.Y., Venayagamoorthy, G.K., Mohagheghi, S., Hemandez, J.-C., Harley, R.G.: Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. IEEE Trans. Evolut. Comput. 12, 171–195 (2008)
17. Chen, X., Li, Y.: A Modified PSO Structure Resulting in High Exploration Ability with Convergence Guaranteed. IEEE Transactions on System, Man and Cybernetics: Part B 37, 1271–1289 (2007)
18. Chen, X., Li, Y.: On Convergence and Parameters Selection of an Improved Particle Swarm Optimization. International Journal of Control, Automation, and Systems 6, 559–570 (2008)
19. Shi, Y., Eberhart, R.: Empirical Study of Particle Swarm Optimization. In: Proc. of the 1999 Congress on Evolutionary Computation, vol. 3, pp. 1945–1950 (1999)
20. Clerc, M., Kennedy, J.: The Particle Swarm – Explosion, Stability, and Convergence in A Multidimensional Complex Space. IEEE Trans. Evolut. Comput. 6, 58–73 (2002)
21. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. IEEE Trans. Evolut. Comput. 10, 281–295 (2006)
22. Zhang, X.X., Chen, W.R., Dai, C.H.: Application of Oriented Search Algorithm in Reactive Power Optimization of Power System. In: Proc. of The Third International Conference on Electric Utility Deregulation and Deregulation and Restructuring and Power Technologies, pp. 2856–2861. IEEE Press, Nanjing (2008)
23. Zhang, X.X., Chen, W.R.: Reactive Power Optimization Based on Oriented Search Algorithm. Journal of Southwest Jiaotong University 45, 418–423 (2010)

Evolution of Cooperation under Social Norms in Non-structured Populations

Qi Xiaowei, Ren Guang, Yue Gin, and Zhang Aiping

Marine engineering of Dalian Maritime university,116026 Dalian, China
xiaowei0735@163.com

Abstract. Indirect reciprocity is a key mechanism for the evolution of human cooperation. There are normally two choices in the standard model of indirect reciprocity which works through reputation. Here we introduced the role of costly punishment into the model. The players could have the third choice besides cooperation and defection. The dynamics of cooperation in indirect reciprocity is analyzed under the social norms which depend on the action of the donor and the reputation of the recipient. It is found that those strategies using costly punishment which allow the evolutionary stability of cooperation typically reduce the average payoff of the population and there is only a small parameter region where costly punishment is evolutionary stable and more efficient. The computer simulations based on agent in finite populations are performed and the result is agreement with our theoretical predictions.

Keywords: costly punishment, indirect reciprocity, dynamics, evolutionary stable strategy, social norm.

1 Introduction

Human societies are organized around cooperative interactions. But why would natural selection equip selfish individuals with altruistic tendencies? This question has fascinated evolutionary biologists for decades. One answer is given in terms of direct reciprocity [1–2]. There are repeated encounters between the same two individuals: I help you, and you help me. More recently, indirect reciprocity has emerged as a more general model: I help you, and somebody helps me. Indirect reciprocity is based on reputation. People monitor the social interactions within their group. Helping others establishes the reputation of being a helpful individual. Natural selection can favor strategies that help those who have helped others [3–4].Direct reciprocity is like an economy based on the exchange of goods, whereas indirect reciprocity resembles the invention of money. In direct reciprocity, one's strategy depends on what his opponent has done to him, but in indirect reciprocity, one's strategy depends not only on what his opponent has done to him but also on others. Direct and indirect reciprocity are two key mechanisms for the evolution of cooperation in human populations.

But in reality, when a person was defected by his opponent, the defector sometimes would be punished by others or his opponent who would not expect benefits from the punishment action and the punisher would pays a cost for exercising punishment.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 347–354, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Axelrod suggested that costly punishment can stabilize social norms[5] and Fehr suggested that costly punishment is an alternative mechanism independent of direct or indirect reciprocity[6]. But the study of the effect of costly punishment on human behavior shows it is not a separate mechanism for the evolution of cooperation but a form of direct or indirect reciprocity. If I punish you because you have defected with me, then I use direct reciprocity. If I punish you because you have defected with others, then indirect reciprocity is at work. In the setting of direct reciprocity, punishment is a form of retaliation [7]. For indirect reciprocity, punishment works through reputation and includes third-party actions, which means that observers of an interaction are willing to punish defectors at a cost to themselves. Therefore, any discussion of the evolution of costly punishment brings us immediately into the framework of direct or indirect reciprocity.

There are many models discussing the direct reciprocity or indirect reciprocity. The repeated Prisoner's dilemma is the most popular form of direct reciprocity. Indirect reciprocity could be described by the variant of TFT[8]. Nowak [9] developed a simple model by image scoring to analyze the dynamics of indirect reciprocity. Here we analyze the dynamics of cooperation in the model based on social norm where the evolutionary programming method is used.

2 Model Summary

Consider a large enough population. At each small time interval, a fraction of players are randomly chosen from the population to form pairs in order to play a game. In each pair, one player acts as a donor and the other player as a recipient. The recipient has a binary reputation, which is either 'good' or 'bad'. The donor has three behavioral choices: cooperation (C), defection (D), and punishment (P). Cooperation involves a cost, c, for the donor and a benefit, b, for the recipient. Defection has no cost and yields no benefit. Punishment has cost α for the donor and cost β for the recipient. The donor's choice depends on the recipient's reputation and his action is observed by other members of the population, who update the donor's reputation according to a social norm, which is shared by all.

After each interaction, the reputation of the donor is updated by the social norms of the population, while the recipient's reputation remains the same. Then the participant goes back to the population with probability ω or leave for ever with probability (1-ω), which makes sure that the total population size remains constant. As a Moran process, the new individual is added in exchange for the lost individual and assigned with good or bad reputation according to the proportion of good and bad players in the current population.

For the social norms, we study them by the second-order assessment[10,11] which depends on the action of the donor and the reputation of the recipient: for example ,it could be deemed as 'good' to cooperate with a good recipient but 'bad' to cooperate with a bad recipient. There are therefore six combinations and 2^6 =64 social norms and 3^2=9 action rules (Fig.1). The action rule specifies the action of the donor when he meets one recipient. For example, CC means that donor will cooperate without considering the recipient's reputation whereas CD means donor would cooperate with a good recipient and defection with a bad one. In contrast with CC and CD, CP action rule prescribes cooperation with good recipients and punishment of bad recipients.

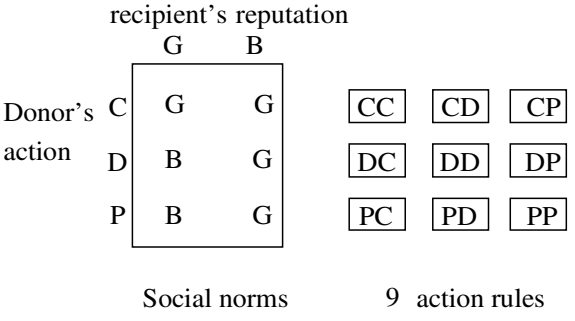


Fig. 1. Social norms and action rules

In the model, there would be errors in updating the reputation after the interaction and it is denoted by μ .With probability (1- μ), the correct reputation is assigned .All individuals share the same conclusions with no private list of reputation.

For notational convenience, the action rule and social norms are defined as $S(*)$ and norm $n(*,*)$. For example, $S(G)=C$ describes an action rule which means when I meet a good recipient , I will take the cooperation. A social norm $n(G,D)=B$ is a mapping from the product of $\{C, D, P\}$ (the donor's action) and $\{G, B\}$ (the recipient's reputation) to $\{G, B\}$ (the donor's new reputation). It suggests that if a donor defect a good recipient, his future reputation will be updated as bad. So under the 64 social norms, we will search for the combination of an action rule $S(*)$ and a social norm n(*,*) ,(S,n), that satisfies the two criteria[14]:

(1) more than 1- μ of all game interactions are cooperative

(2) under social norm n, the action rule S is evolutionarily stable

The dynamic programming equation (Bellman equation) is used to study the evolutionary stability of the action rule S. It is assumed that all players except the focal player adopt action rule S under the given social norm $n(*)$.If the best-response action rule S_{opt} exists uniquely and matches S, then S is evolutionarily stable.

If the strategy is the best response to itself, it would get the maximum payoff when interacting with others.

$$W_{I,J} = \max_{X=C,D,P}[\frac{1}{2}\{-\xi[X]+\omega W_{(1-\mu)n(J,X)+\mu\overline{n(J,X)},(1-\mu)G+\mu B}\}+\frac{1}{2}\{\eta[s(I)]+\omega W_{I,(1-\mu)G+\mu B}\}] \quad (1)$$

Where W_{IJ} describes the expected maximum payoff that a player, currently having reputation $I(=G$ or $B)$,can gain from the game with his opponent with reputation $J(=G$ or $B)$ from nowadays to future. $\xi[X]$ and $\eta[X]$ is respectively the cost and benefit function of action X.

It is obviously that the W_{IJ} is recurrence and we have to make some mathematical skills for solving the equation. The equation (1) could be rewritten as equation (2) because of W_{IJ} independents of I :

$$Sopt(J) = \arg\max_{X=C,D,P}\{\omega W_{(1-\mu)n(J,X)+\mu\overline{n(J,X)},(1-\mu)G+\mu B} + \xi[X]\} \quad (2)$$

where $\phi_G[G]=1$, $\phi_G[B]=0$ and $q=1-2\mu$ .The $q=1-2\mu$ is also called 'social resolution' which defines the ability to distinguish between good and bad and is a key parameter in indirect reciprocity.

With the item $\frac{\omega}{2}W_{(1-\mu)B+\mu G,(1-\mu)G+\mu B}$ being constant, so it is could be ignored in the calculation. The first term, $-\xi[X]$, represents the immediate cost of action *X*. The second term $\omega q\lambda\phi_a[n(J,X)]$ represents the future benefit through becoming a good player via action *X*, which is $\lambda\phi_a[n(J,X)]$ multiplied by the discounting factor ω and the social resolution q . Hence we are able to derive *s**(*G*) and *s**(*B*). By using the method described above, we get the four kind of combination as showed in figure 2.. In the first combination, where the cost of punishment is smaller than cost of cooperation ($c>\alpha$), the CD action rule is ESS under the condition $\omega qb>(2-\omega)c$ and the average payoff is $\frac{1}{2}[(1-\mu)\frac{b-c}{1-\omega}]$.

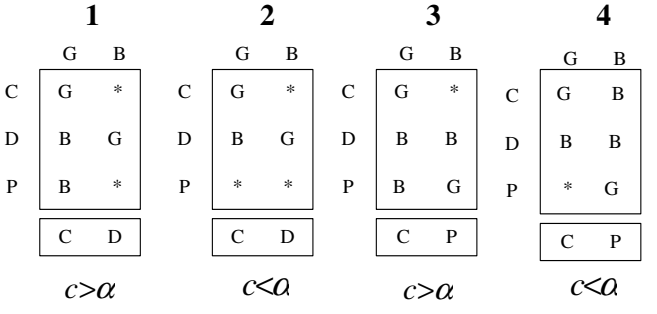


Fig. 2. Social norms of cooperation

3 Simulation Results and Analysis

To confirm the analytic predictions, we have run agent-based computer simulations where the size of population is a fixed number, N = 100. In the beginning, each new player receives an initial reputation, which is either good or bad with equal probability. Each player adopts one of 9 possible action rules. All players share the same social norm that is fixed in the population. In each step of updating, every individual has exactly 10 interactions with other randomly chosen individuals. Individuals play donor and recipient on average 5 times at each interaction. Then the reputation of the donor is updated according to the social norm, but a wrong reputation is assigned with probability μ = 0.02. In this case, every player agrees on the wrong reputation of this particular player. No private lists of reputation are considered. In the latter, we will release the limit. After all interactions have taken place, an individual is chosen for reproduction with a probability proportional to P_i-P_{min}, where P_i is the total payoff of the individual and P_{min} is the minimum payoff in the population. The offspring inherits the action rule of the parent and replaces

another random player. However, the offspring sometimes will randomly choose one of the 9 action rules as his own strategy with probability ε = 0.01. It could be understood in another way that error rate in assigning reputation of offspring is ε. After reproduction, the payoffs of all players are reset to zero. Thus, older players do not accumulate their payoffs. The total generations of the simulations are 1000.

In the simulation, there are two classic norms for demonstration: stern-judging, and shunning [12]. First, we do the simulations with no error(represented as eact) in executing intended action and error(represented as erep) in recalling recipient's reputation.

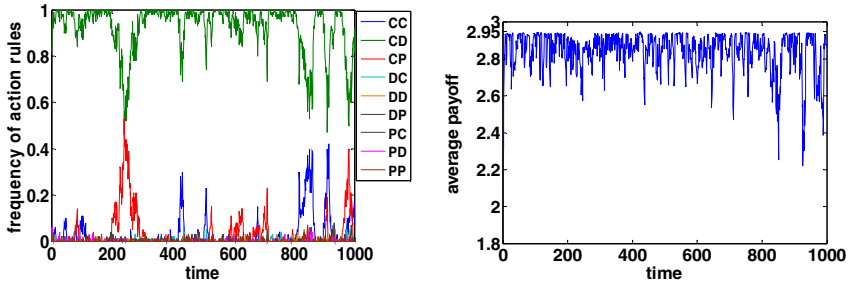


Fig. 3. Simulation results for b> c in stern-judging
b=9, c=3, a=1, β =4, μ=0.02,ε = 0.01 , e_{act} =0 and $e_{rep=}$ 0

In figure 3, the parameter values are: b = 9, c = 3, α = 1, β = 4, μ = 0.02, eact =0 and erep= 0 and ε = 0.01. The initial frequency of the CD action rule is 0.8. The initial frequencies of the other 8 action rules are randomly chosen .We find that the CD(green) action rule is stable against invasion attempts by CC (blue) and CP (red). The average payoff of the population per round fluctuates with its maximum being 2.94, which agrees with our analytic prediction, (1 − μ)(b − c)/2 = 2.94.

In figure 4, the initial frequency of CP is 0.8. CP is stable against other action rules. The average payoff per round fluctuates with its maximum being 2.89, which also agrees with our analytic prediction, {(1 − μ)(b − c) − μ(α+ β)}/2 = 2.89.

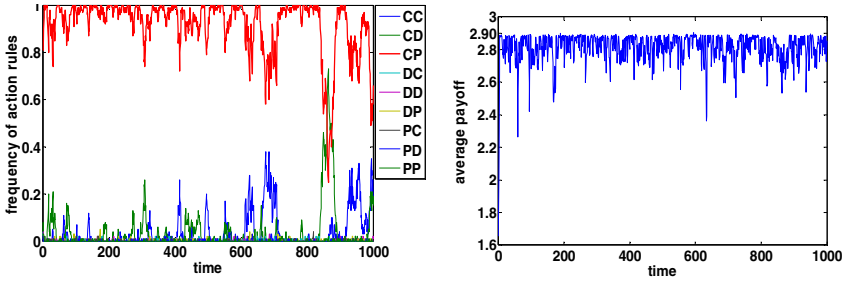


Fig. 4. Simulation results for b> c in shunning
b=9, c=3, a=1, β =4, μ=0.02,ε = 0.01 , eact =0 and erep= 0

In figure 5,when b<c in stern-judging, the CD(green) action rule, whose initial frequency is 0.8, is immediately invaded by DD. In this parameter region, the CD rule is not evolutionarily stable .The average payoff is around 0. So DD would be a more profitable action rule for the group when b<c.

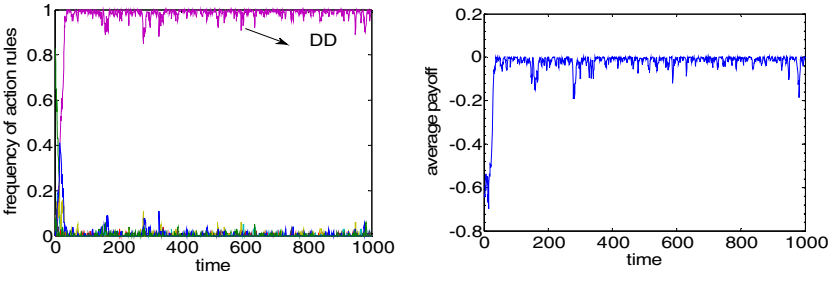


Fig. 5. Simulation results for b< c in stern-judging
b=2, c=3, a=1, β =4, μ=0.02,ε = 0.01 , eact =0 and erep= 0

In figure 6, the CP action rule is robust against invasion by the other 8 action rules because of the large effect of punishment (β = 4) although cooperation is non-productive because b < c. The average payoff is negative and is about -0.54. Hence the group does not benefit from the cooperation that is enforced by costly punishment.

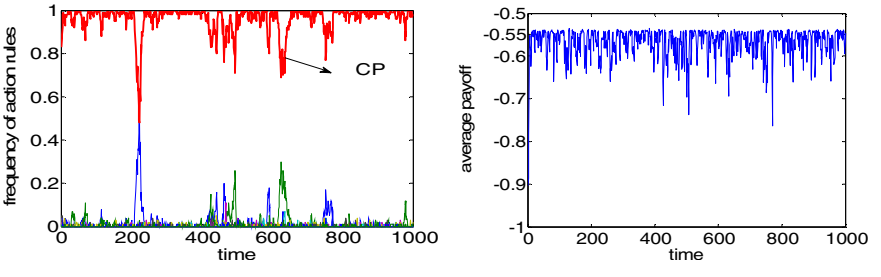


Fig. 6. Simulation results for b< c in shunning
b=2, c=3, a=1, β =4, μ=0.02,ε = 0.01 , eact =0 and erep= 0

Here, we take computer simulation to analyze the impact of the initial frequency and error (μ) in assigning reputation on the cooperation. In the following table, there is the necessary round number for stability in different norm. For example, in stern-judging norm, if initial frequency of CD is 0.1, μ=0.02, the necessary round number for the population to maintain cooperation is 120.

The round number needed in shunning norm is less than Stern-judging norm. Probably the reason is that the ESS condition of CD is more rigorous than CD. When the error μ is less than 0.3, the most population will take the choice of cooperation if the communication round number is more than 400.But if the error μ is bigger, the population finally will be selfish no matter how big the communication round number.

Table 1. Round number of communication in Stern-judging

CD	round μ=0.01	round μ=0.03	round μ=0.05	round μ=0.07	round μ=0.1	round μ=0.2
0.1	110	120	380	350	500	>500
0.3	100	160	240	350	480	>500
0.5	100	160	220	350	490	>500
0.7	90	170	220	340	460	>500

Table 2. Round number of communication in shunning

CP	round μ=0.01	round μ=0.03	round μ=0.05	round μ=0.07	round μ=0.1	round μ=0.2
0.1	50	50	150	170	210	>500
0.3	50	80	110	130	210	>500
0.5	50	50	90	150	230	>500
0.7	60	50	110	130	210	>500

If we extend the mechanism to group communication that a group of people interact with each other to exchange their message, the result may be better. Furthermore, if the group is the whole population, then there is no private list and no error in assigning reputation. To get cooperation, the round number has to increase as the total size of the population increases.

4 Conclusion

We have studied the dynamics of indirect reciprocity in an explicit model and analyzed all social norms that use binary reputation and second-order assessment. We find that both CD and CP action rules can stabilize cooperation. These rules reward good recipients with cooperation and 'punish' bad ones with either defection (CD) or costly punishment (CP). If both CD and CP action rules are evolutionarily stable, the use of costly punishment leads to a lower equilibrium payoff and is therefore inefficient. It is even possible that costly punishment yields a lower payoff than all defection (DD). Costly punishment maximizes the group average payoff for only a very limited parameter region. This narrow margin of efficiency requires fine-tuning of the key parameters. If the social resolution exceeds the cost to benefit ratio (q>c/b), which is the same as Hamilton rule in non-relatives, CD rules are always more efficient than CP rules.

If the reputation system lost or weaken, the cooperation would be at low-level. However, there are some phenomena which show the tradition virtue is threatened. In Chongqing, china, a pupil helps an old lady up, but he was charged with pushing the lady down. If people all are afraid of being in trouble with helping others, how will the society be? Now our country is in the process of Social Transformation, the society needs harmonious and stable socio-economic environment. The government should do their best to induct people to keep the traditional virtue.

Furthermore, in this paper, it is assumed that there is only one best response strategy under the social norm, but the simulation results reveal that two strategies could coexist. So it might be worthwhile to analyze the dynamics in the population on the assumption that there are two best response strategies. Maybe there would be some much more interesting findings.

References

1. Nowak, M.A., Sigmund, K.: Evolution of indirect reciprocity by image scoring. J. Nature 393, 573–577 (1998)
2. Dufwenberg, M., Gneezy, U., Güth, W., van Damme, E.: Direct vs indirect reciprocity: an experiment. J. Homo. Oecon. 18, 19–30 (2000)
3. Brandt, H., Sigmund, K.: The logic of reprobation: assessment and action rules for indirect reciprocation. J. Theor. Biol. 213, 475–486 (2004)
4. Bolton, G.E., Katok, E., Ockenfels, A.: Cooperation among strangers with limited information about reputation. J. Public Econ. 89, 1457–1468 (2005)
5. Alexander, R.C.: The Biology of Moral Systems. Aldine de Gruyter, New York (1988)
6. Fehr, E., Gächter, S.: Altruistic punishment in humans. J. Nature 415, 137–140 (2002)
7. Yamagishi, T.: Seriousness of social dilemmas and the provision of a sanctioning system. J. Social Psychology Quarterly 51, 32–42 (1988)
8. Clutton-Brock, T.H., Parker, G.A.: Punishment in animal societies. J. Nature 373, 209–216 (1995)
9. Fehr, E., Gächter, S.: Altruistic punishment in humans. J. Nature 415, 137–140 (2002)
10. Fehr, E., Fischbacher, U.: Third-party punishment and social norms. J. Evol. Hum. Behav. 25, 63–87 (2004)
11. Owler, J.H.: Altruistic punishment and the origin of cooperation. J. Proc. Natl. Acad. Sci. USA. 102, 7047–7049 (2005)
12. Ohtsuki, H., Iwasa, Y.: The leading eight: Social norms that can maintain cooperation by indirect reciprocity. J.Theoretical Biology 239, 435–444 (2006)
13. Suzuki, S., Akiyama, E.: Chaos, oscillation and the evolution of indirect reciprocity in n-person games. J.Theoretical Biology 252, 686–693 (2005)

Collaborative Optimization under a Control Framework for ATSP

Jie Bai[1], Jun Zhu[2], Gen-Ke Yang[1], and Chang-Chun Pan[1]

[1] Automation Department of Shanghai Jiao Tong University, 200240, Shanghai, China
[2] Jiangsu Jun-Long Electrical Technology CO., Ltd.
pan_cc@sjtu.edu.cn

Abstract. A collaborative optimization algorithm under a control framework is developed for the asymmetric traveling salesman problem (ATSP). The collaborative approach is not just a simple combination of two methods, but a deep collaboration in a manner like the feedback control. A notable feature of the approach is to make use of the collaboration to reduce the search space while maintaining the optimality. Compared with the previous work of the reduction procedure by Carpaneto, Dell'Amico et al. (1995) we designed a tighter and more generalized reduction procedure to make the collaborative method more powerful. Computational experiments on benchmark problems are given to exemplify the approach.

Keywords: ATSP, Collaborative optimization, Control framework, Ant colony optimization, Branch and Bound.

1 Introduction

ATSP is one of the most well-known combinatorial optimization problems due both to its practical relevance and to its considerable difficulty. **ATSP** and its variations are commonly used models to formulate many practical applications, such as the scheduling of chemical process[3], the scheduling of steel production[17][18], and printed circuit board punching sequence problem [15], and so on. The problem is concerned with finding the shortest Hamiltonian cycle or tour in a weighted directed graph without loops and multiple arcs. Although simple to state, **ATSP** is very difficult to attack and much effort has been, and will continue to be, devoted to the design of good optimization algorithms. Roughly speaking, methods of solving **ATSP** problems can fall into three categories, i.e., rigorous, heuristic and hybridized. Rigorous method can guarantee the optimality of the solution obtained. Many algorithms have been developed for the exact solution of **ATSP**. The representative ones are the Branch-and-Bound (**B&B**) based on Assignment Problem (**AP**) relaxation[2][16]. Recently, Turkensteen, Ghosh et al.[19] reported a new branching rule in **B&B** algorithm which utilizes ***tolerances*** to indicate which arcs are preferred to save in the optimal **ATSP** tour. The Branch-and-Cut (**B&C**) algorithm is also explored by [8]. Since solving the **ATSP** optimally is **NP**-hard, especially in many real industrial problems, exact optimization algorithms require overlong execution

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 355–363, 2011.
© Springer-Verlag Berlin Heidelberg 2011

time and huge memory. In most of cases they can't produce an acceptable or even feasible solution given limit time. Hence, heuristics were dominating later even though they can't provide any guarantee on the solution quality. Karp [13] presented the so-called Patching Algorithm **(PA)-**a convincible heuristic method, and showed that the heuristic solution asymptotically converges to the optimal solution as the size of **ATSP** tends to infinity. Glover, Gutin et al. [9] introduced several construction heuristics, and conducted a large number of computational experiments for several families of **ATSP** instances. In more recent years, metaheuristics are booming, such as genetic algorithms (**GA**) [3][10], simulated annealing **(SA)** [14], tabu search **(TS)** [7], ant colony optimization **(ACO)**[1][6][15], and so on. Rigorous and heuristic approaches have ever-conflicting advantages and disadvantages in terms of computational load and solution quality. Hence, growing attentions have been given to hybridized methods.

The rest of the paper is organized as follows. In the next section, the mathematical formulation of the **ATSP** is presented. In section 3, the collaborative optimization framework is described. The key components that constitute the framework are illustrated in detail. Furthermore, theoretical analysis with respect to the performance is also presented briefly. In section 4, a large number of computational experiments are given. Finally, concluding remarks are included in section 5.

2 Problem Description

The **ATSP** can be formulated as an Integer Linear Programming

$$z^* = \min \sum_{(i,j)\in \boldsymbol{A}} c_{(i,j)} \cdot x_{(i,j)} \tag{1}$$

subject to:

$$\sum_{(i,j)\in \boldsymbol{A}} x_{(i,j)} = 1 \quad i = 1,\cdots,n \tag{2}$$

$$\sum_{(i,j)\in \boldsymbol{A}} x_{(i,j)} = 1 \quad j = 1,\cdots,n \tag{3}$$

$$\sum_{i\in \boldsymbol{S}} \sum_{j\in \boldsymbol{S}} x_{(i,j)} \le |\boldsymbol{S}| - 1 \qquad \boldsymbol{S} \subset \{1,2,\cdots,n\}, 2 \le |\boldsymbol{S}| \le n-2 \tag{4}$$

$$x_{(i,j)} \in \boldsymbol{D}_{(i,j)} \triangleq \{0,1\} \qquad (i,j)\in \boldsymbol{A} \tag{5}$$

where $x_{(i,j)} = 1$ $(i,j)\in \boldsymbol{A}$, if the arc is selected in the optimal solution; and $x_{(i,j)} = 0$ otherwise. Apparently, equations (1)(2)(3) and (5) define the **AP** problem and constraints (4) forbid subtours. The number of constraints described in constraints (4) will be exponentially explosive as the size of the problem increases. This reason cause the computational difficult in solving large-scale **ATSP**.

3 The Collaborative Optimization

3.1 The Overall Scheme

Commonly, the **ATSP** solution space $\boldsymbol{D}$ can be described by

$$\boldsymbol{D} = \prod_{(i,j)\in \boldsymbol{A}} \boldsymbol{D}_{(i,j)} \ . \ \boldsymbol{D}_{(i,j)} = \{0,1\} \tag{6}$$

Once some particular arc (i, j) is indentified definitely not belonging to optimal solutions which means we can fix $x_{(i,j)} = 0$, the entire search space $\boldsymbol{D}$ would shrink by 50%. Based on the result, we propose a parallel optimization architecture in which the exact solver and meta-heuristic solver are launched concurrently (see Fig1).

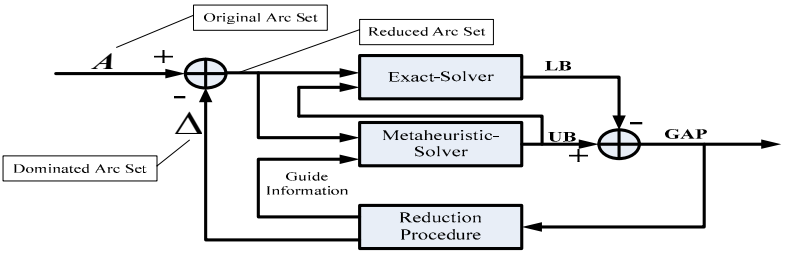


Fig. 1. A generalized parallel optimization architecture for ATSP where The *dominated arc set* in this paper is referred to as those arcs that not belong to optimal solutions or those ones if included in a solution that can't generate better solution than the best-known one so far

Fig.1 mainly consists of three components:

- Exact solver: the exact solver works through iteratively solving simplified subproblems and thus it can provide a lower bound (**LB**) (a minimum problem).
- Metaheuristic Solver: the metaheuristic solver usually works by generating feasible solutions and then conducting improvement. The objective of the best solution can be adopted as an upper bound (**UB**).
- Reduction Procedure: the reduction procedure takes useful information from both exact and metaheuristic solver to reduce the global search space and hence speed up the rate of convergence to optimality.

3.2 The Implementation

A more concrete optimization scheme is constructed in which **B&B** method and **ACO** are used respectively (see Fig. 2). PATCH is referred to as the Patch Algorithm (**PA**) proposed by [13] which is employed as another upper bounding strategy. As shown in the diagram, ***B&B***, ***Reduction Procedure***, ***Patching Algorithm*** and ***ACO*** are 4 key components that compose the framework. The details will be specified in the following sections.

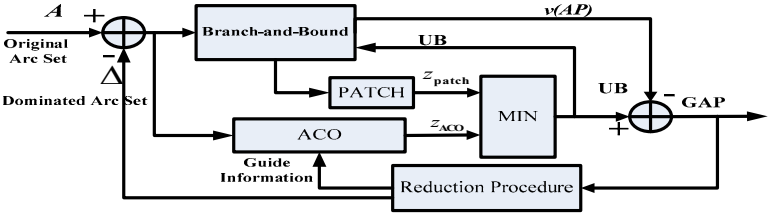


Fig. 2. A concrete implementation for **ATSP**. $z_{\mathbf{patch}}$ and $z_{\mathbf{ACO}}$ are the objective value produced by **PA** and **ACO** respectively**.**

A. B&B method

With **B&B** method we use the ***breadth-first*** search instead of the ***lowest-first*** in [2]. The advantage is that the low bound obtained would be better and better as the approach progresses since our intention is to provide a high quality solution with the guaranteed degree of accuracy.

B. *Reduction procedure*

According to the dual theory of **LP**, the dual problem of **AP** is defined as follows

$$v(\mathbf{AP}) = \max\left(\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j\right)$$

Subject to:

$$u_i + v_j \le c_{(i,j)}$$

Carpaneto, Dell'Amico et al. [2] based on the dual rpoblem presented a reduction procedure as lemma 1 shows

Lemma 1. If a feasible solution (**ATSP**) with objective value z **(UB)** is known, then each $(i,j) \in \boldsymbol{A}$ with

$$\overline{c}_{(i,j)} \ge z - v(\mathbf{AP}) \quad or \quad \overline{c}_{(i,j)} + v(\mathbf{AP}) \ge z \tag{7}$$

can be discarded from further considerations.
The above result is further strengthened by the theorem 1

Theorem 1. if arc (i,j) satisfies the following property

$$v(\mathbf{AP}) + \overline{c}_{(i,j)} + \overline{d}\left(\boldsymbol{col}^{\bullet}[j], \boldsymbol{row}^{\bullet}[i]\right) \ge z \tag{8}$$

The arc (i,j) can be ignored in the further optimization procure.

where $\boldsymbol{col}^{\bullet}$ and $\boldsymbol{row}^{\bullet}$ are the optimal solution to the initial **AP** corresponding to the row and column respectively (row and column are commonly used notations in the context of **AP** and $k = \boldsymbol{col}^{\bullet}[j]$ means row k is assigned to column j which indicates that arc (k,j) is belong to some certain subtour in the context of **ATSP**.;

$\bar{d}\left(\boldsymbol{col}^{\bullet}[j],\boldsymbol{row}^{\bullet}[i]\right)$ is called as the shortest path cost on the newly defined bipartite digraph $\bar{\boldsymbol{G}}=\left(\boldsymbol{U}\cup\boldsymbol{V},\tilde{\boldsymbol{D}}\cup\tilde{\boldsymbol{R}}\right)$ whose definition is given in the proof procedure.

Proof. The definition of the bipartite digraph, i.e., $\bar{\boldsymbol{G}}$ is first constructed. Then $\boldsymbol{U},\boldsymbol{V}$ are the row and column node sets, $\boldsymbol{U}=\boldsymbol{V}\triangleq\{1,2,\cdots,n\}$ .Denote $k=\boldsymbol{col}^{\bullet}[j],k\in\boldsymbol{U},j\in\boldsymbol{V}$ and $l=\boldsymbol{row}^{\bullet}[i],i\in\boldsymbol{U},l\in\boldsymbol{V}$ as the best assignment node for node j and i respectively. Then the optimal solution $\boldsymbol{X}$ can be represented as the following form:

$$\boldsymbol{X}=\left\{(i,j)\mid j=\boldsymbol{row}^{\bullet}[i]\ or\ i=\boldsymbol{col}^{\bullet}[j],i\in\boldsymbol{U},j\in\boldsymbol{V}\right\}$$

the arc set $\tilde{\boldsymbol{D}}\cup\tilde{\boldsymbol{R}}$ is defined as below:

$\tilde{\boldsymbol{D}}$ is the direct arcs

$$\tilde{\boldsymbol{D}}=\left\{(g,h)\mid g\in\boldsymbol{U},h\in\boldsymbol{V},(g,h)\in\boldsymbol{A}\right\}\setminus\boldsymbol{X}$$

$$\bar{c}_{(g,h)}=c_{(g,h)}-u_g-v_h,\quad (g,h)\in\tilde{\boldsymbol{D}}$$

$\tilde{\boldsymbol{R}}$ is the reverse arcs

$$\tilde{\boldsymbol{R}}=\left\{(h,g)\mid h\in\boldsymbol{V},g\in\boldsymbol{U},(g,h)\in\boldsymbol{X}\setminus\left\{(k,j),(i,l)\right\}\right\}$$

$$\bar{c}_{h,g}=0,(h,g)\in\tilde{\boldsymbol{R}}$$

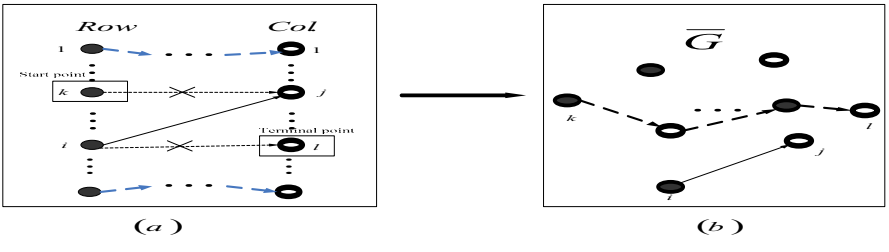


Fig. 3. A graphical illustration for computing a lower-bound on the cost of ATSP including (i,j)

Fig 3 depicts a illustration for computing a lower-bound on the cost of **ATSP** solution including arc (i,j) . If (i,j) is definitely required in the **AP** solution, then the matching pairs (k,j) and (i,l) in the $\boldsymbol{X}$ have to be broken, such that the node $k\in\boldsymbol{U}$ is a unassigned row node, and $l\in\boldsymbol{V}$ is a unassigned column node. In order to get a new feasible optimal solution $\boldsymbol{X}'$, one must find the shortest alternating path from node k to node l in $\bar{\boldsymbol{G}}$ [5]. It is obvious that any dipath of $\bar{\boldsymbol{G}}$ starting from a node of $\boldsymbol{U}$ and terminated with a node of $\boldsymbol{V}$ contains, alternatively, an arc in $\boldsymbol{D}$ and an arc in $\boldsymbol{R}$. Such that cost of the new optimal match $\boldsymbol{X}'$ is exactly :

$$v(\boldsymbol{X}') = v(\boldsymbol{X}) + \bar{c}_{(i,j)} + \bar{d}(k,l) = v(\mathbf{AP}) + \bar{c}_{(i,j)} + \bar{d}(k,l) \tag{9}$$

Eq.(9) provides a tighter lower bound on the cost of **ATSP** that (i, j) is required. So, if (8) is satisfied, it means that no **ATSP** solutions with the inclusion of arc (i, j) is better than z.

Theorem 1 tells the effective way to filter out dominated arcs while for the remaining arcs we have the following theorem which actually is a generalization of lemma 1.

Theorem 2. Given a known objective value z, any feasible solution that contains the following *K* arcs can't be better than z if the following inequality holds

$$\sum_{k=\{1,\cdots,K\}} \bar{c}_{(i_k,j_k)} + v(\boldsymbol{AP}) \geq z \tag{10}$$

Proof. This result is an immediate result inspired by inequality (7).

Theorem 2 is very useful for guiding **ACO**. In the construction phase of the algorithm, an ant incrementally constructs a complete tour by adding nodes to a partial path, say s^p, constructed so far. Hence, **Theorem 2** provides a good way to assess the possible obtained tour starting with the partial path s^p. This is done by calculating the accumulated reduced cost of s^p and check whether it excess the gap between $v(\boldsymbol{AP})$ and z. If it is the case by which we mean any tour containing s^p is worse than the obtained solution with the objective value z, such that the ant will terminate the search and return to the starting point. This mechanism will increase the efficiency of **ACO**.

Moreover, based on theorem 2 the transition probability can be redesigned as follows:

Denote $\boldsymbol{\Phi}$ as the set of cities that should be priced out

$$\boldsymbol{\Phi} = \left\{k \mid \eta + \bar{c}_{l,k} \geq z - v(\mathbf{AP}), k \in \boldsymbol{V} \setminus s^p\right\} \tag{11}$$

where $\eta = \sum_{(i,j)\in s^p} \bar{c}_{ij}$ is the accumulative reduced cost of s^p and then

$$P\left(l \mid r \in s^p\right) = \begin{cases} \dfrac{\tau(r,l)\cdot\eta(r,l)^{\beta}}{\sum_{j\in \boldsymbol{V}\setminus\left\{\boldsymbol{\Phi}\cup s^p\right\} \text{ and } (r,j)\in \boldsymbol{A}\setminus\boldsymbol{\Delta}} \tau(r,j)\cdot\eta(r,j)^{\beta}} & \text{if } l \in \boldsymbol{V} \setminus\left\{\boldsymbol{\Phi}\cup s^p\right\} \\ 0 \quad \text{otherwise} \end{cases} \tag{12}$$

The reduction procedure inevitably requires additional computing cost. The question is whether the reduction in the search space is sufficiently large to compensate for the additional computing cost. Computational experiments show that it really works for most of benchmark problems.

4 Computational Experiments

The computational experiments are executed on a **1.79GHz** personal computer and totally 19 benchmark problems are tested. The proposed algorithm is coded in **C++**. Although many well-crafted algorithms have been designed for **ATSP**, we don't intend giving extensive comparisons among all those algorithms. Since the proposed optimization framework is to exhibit the power of collaboration between exact methods and metaheuristics. So, the comparisons will be among our method, **ACO**[6] and **CDT**[2]. This is to give a direct evidence to verify that the proposed method outperforms the methods that works individually.

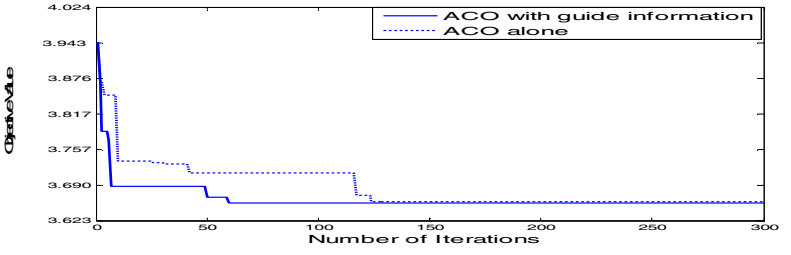


Fig. 4. The performance comparison of two ACO algorithms for instance 'kro124'

The parameters of **ACO** are set to the same values as the literature [6] did. In the implementation, the local search, e.g. 2-opt or 3-opt, is not used in **ACO** in order to illustrate directly the efficacy of collaborations between methods. Fig.4 presents comparative convergent curves for the two kinds of **ACO** algorithms. The proposed algorithm has faster convergent rate and higher solution quality. Table 1 summarizes

Table 1. Comparison results of several existing algorithms

Instance	Known Best	Time for hitting the best known (sec.)		
		ACO alone	CDT	Our method
br17	39	**0.001**	**0.001**	**0.001**
ft53	6905	24	--	**20**
ft70	38673	9	67	**2**
ftv33	1286	10	**1**	2
ftv35	1473	8	4	**3**
ftv38	1530	8	**3**	5
ftv44	1613	5	7	**3**
ftv47	1776	16	10	**5**
ftv55	1608	17	7	**3**
ftv64	1839	12	**4**	6
ftv70	1950	25	127	**23**
ftv170	2755	105	--	**53**
kro124	36230	150	--	**89**
p43	5620	**35**	--	47
ry48p	14422	23	12379	**13**
rbg323	1326	37.43	**0.001**	**0.001**
rbg358	1163	105.32	**0.001**	**0.001**
rgb403	2465	24.39	**0.001**	**0.001**
rgb443	2720	65.63	**0.001**	**0.001**

Note that all the experiments are starting with the solution obtained by **PA. "**--"means the best known solution is not found in 10 hours by **CDT** algorithm and the number marked in bold style indicates the best behavior in terms of running time.

the results among the algorithms. See that for 80% instances the proposed algorithm is better than the other two.

5 Conclusion

A novel collaboration optimization framework is proposed to solve the **ATSP**. The advantage mainly lies in the synergistic utilization of interior information which improves the overall searching efficacy. The idea behind is that the proposed algorithm is powerful in dealing with **ATSP** because it is equipped with the capacity to exploit the interior information concerning the characteristics of instances they are solving. The design of collaborative algorithms combing rigorous method and metaheuristics will continue to be one of the most promising research areas in discrete optimization.

Acknowledgements

The research was supported by National Natural Science Foundation of China (Grant 61074150).

References

1. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 34(2), 1161–1172 (2004)
2. Carpaneto, G., Dell'Amico, M., et al.: Exact solution of large-scale, asymmetric traveling salesman problems. ACM Trans. Math. Softw. 21(4), 394–409 (1995)
3. Choi, I.-C., Kim, S.-I., et al.: A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. Comput. Oper. Res. 30(5), 773–786 (2003)
4. Choi, J., Realff, M.J., et al.: An algorithmic framework for improving heuristic solutions Part I. A deterministic discount coupon traveling salesman problem Computers & Chemical Engineering 28(1): 1285-1296 (2004). Computers & Chemical Engineering 28(1), 1285–1296 (2004)
5. Dell'Amico, M., Toth, P.: Algorithms and codes for dense assignment problems: the state of the art. Discrete Applied Mathematics 100(1-2), 17–48 (2000)
6. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
7. Fiechter, C.N.: A parallel tabu search algorithm for large traveling salesman problems. Discrete Appl. Math. 51(3), 243–267 (1994)
8. Fischetti, M., Toth, P.: A Polyhedral Approach to the Asymmetric Traveling Salesman Problem. Management Science 43(11), 1520–1536 (1997)
9. Glover, F., Gutin, G., et al.: Construction heuristics for the asymmetric TSP. European Journal of Operational Research 129(3), 555–568 (2001)
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)

11. John, D.L.: An improved solution to the traveling salesman problem with thousands of nodes. Commun. ACM 27(12), 1227–1236 (1984)
12. Jourdan, L., Basseur, M., et al.: Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research 199(3), 620–629 (2009)
13. Karp, R.M.: A Patching Algorithm for the Nonsymmetric Traveling-Salesman Problem. SIAM Journal on Computing 8(4), 561–573 (1979)
14. Laarhoven, P.J.v., Aarts, E.H.: Simulated Annealing: Theory and Applications. Mathematics and Its Applications. Springer, Heidelberg (1987)
15. Marco, D., Christian, B.: Ant colony optimization theory: a survey. Theor. Comput. Sci. 344(2-3), 243–278 (2005)
16. Miller, D.L., Pekny, J.F.: Exact Solution of Large Asymmetric Traveling Salesman Problems. Science 251(4995), 754–761 (1991)
17. Pan, C., Yang, G.K.: A method of solving a large-scale rolling batch scheduling problem in steel production using a variant of column generation. Comput. Ind. Eng. 56(1), 165–178 (2009)
18. Tang, L.X., Liu, J.Y., et al.: A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. European Journal of Operational Research 124(2), 267–282 (2000)
19. Turkensteen, M., Ghosh, D., et al.: Tolerance-based Branch and Bound algorithms for the ATSP. European Journal of Operational Research 189(3), 775–788 (2008)

Bio-Inspired Dynamic Composition and Reconfiguration of Service-Oriented Internetware Systems

Huan Zhou[1], Zili Zhang[1,2,⋆], Yuheng Wu[1], and Tao Qian[1]

[1] Faculty of Computer and Information Science
Southwest University, Chongqing 400715, China
[2] School of Information Technology, Deakin University, VIC 3217, Australia
zhangzl@swu.edu.cn

Abstract. Dynamic composition and reconfiguration of service-oriented Internetware systems are of paramount importance as we can not pre-define everything during the design time of a software system. Recent biology studies show that the slime mold *Physarum polycephalum* – a single-cell organism – can form a veined network that explores the available space and connects food sources in the absence of central control mechanisms. Inspired by the formation and behavior of such biological adaptive networks, a new bionic approach is proposed for dynamic service composition and reconfiguration of Internetware systems. Simulation experiments were conducted. The experimental results show that the proposed approach is effective and efficient. It is hoped that this paper will shed new light in Internetware system design and construction.

Keywords: Internetware, Physarum polycephalum, Service Composition, Reconfiguration.

1 Introduction

Any service-oriented software system running in an open and dynamic environment like the Internet can be called an Internetware system. To meet the functional and quality requirements, the dynamic composition and reconfiguration capabilities are required for such systems. With the dynamic composition and reconfiguration capabilities, the Internetware systems can improve and optimize their configurations when the network environment changes. An Internetware system provides comprehensive services by combining various service components. It is evident that we cannot pre-define everything for such a software system, and some unknown emergent behaviours may occur when it is running under a changing network environment. A service component failure or network congestion may result in the system failure or unreliable results. If this happens, redeveloping and redeploying a new system will cost too much. A better way

⋆ Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 364–373, 2011.
© Springer-Verlag Berlin Heidelberg 2011

is to reconfigure the system dynamically during the running time to adapt the changed environment or tasks.

Dynamic service composition and reconfiguration of service-oriented software systems have attracted much attention in the past few years. Typically, there are two principal approaches for dynamic service composition – process-driven and semantic-driven [1]. The process-driven approach includes the workflow model method [2], the state calculus approach [3], the process algebra approach [4], and so on. The main idea of semantic-driven approach is to introduce machine understood semantics for service description and service request. In this way, the composition schemes can be generated automatically through reasoning [5].

Recent years also saw many studies on reconfiguration. In [6], Ribeiro-Justo et al discussed the integration of $FRODICA$ (Framework for Distributed Configurable Applications) framework and $CODA$ (Complex Organic Distributed Architecture) framework. A multi-role and multi-layered framework is then formed to monitor and reconfigure a system. Anosike et al proposed a solution for the logistics reconfiguration problem in manufacture sectors, in which the negotiation and bidding of multi-agents without centralized control is utilized [7]. The self-organization of distributed multi-agent systems was achieved through negotiation and trust management mechanisms [8]. Approaches were proposed to solve the incremental web services publication and registration problem in composition Web services through service community, composite services, atomic services, and the heterogeneous hierarchy of service providers [9]. The theoretical foundation was also laid to assure the correct operation of composition services. Palma et al [10] proposed an agent system reconfiguration approach by modifying the geographic distribution of agents in the system, which is based on a structural description language called ADL.

It is noted that many organisms can achieve self organization and self optimization without global information and centralized control. Borrowing ideas from biology has helped us to successfully develop neural network algorithms, genetic algorithms and ant colony optimization algorithms etc. A recent study shows that the slim mold *Physarum polycephalum* can demonstrate self organization, self optimization and self repair behaviours naturally [11]. Its foraging behavior could guide people to improve technical systems, such as mobile communication network or other dynamic networks of connected computing devices. Drawing inspiration from this, we propose a novel dynamic service composition and reconfiguration approach for Internetware systems in this paper. It is hoped that this paper will shed new light for the design and construction of Internetware systems.

The rest of the paper is organized as follows. In Section 2, background knowledge of the foraging principles of *Physarum polycephalum* and its corresponding mathematical model as well as Internetware is introduced. Section 3 presents the bio-inspired service composition and reconfiguration model. Experimental results and discussions are provided in Section 4. Section 5 concludes the paper.

2 Background

For the sake of further discussion, a brief introduction to the foraging principle of *Physarum polycephalum* and its corresponding mathematical model as well as Internetware is provided in this section.

2.1 Foraging Principles of *Physarum polycephalum* and Its Mathematical Model

The slime mould *Physarum polycephalum* is a kind of large amoeba-like cell consisting of a dendritic network of tube-like structures (pseudopodia). It changes its shape as it crawls over a plain agar gel and, if food is placed at two different points, it will put out pseudopodia that connects the two food sources in order to absorb nutrition through the flow of protoplasm. At last, an optimized network is formed by connecting different food sources.

The physiological mechanism of tube formation can be summarized as follows: Tubes thicken in a given direction when shuttle streaming of the protoplasm persists in that direction for a certain time. This implies positive feedback between flux and tube thickness, as the conductivity of the sol is greater in a thicker tube. Experimental results have revealed two empirical rules describing the changes in the tubular structure of the plasmodium: first, open-ended tubes are likely to disappear; and second, when two or more tubes connect the same two food sources, the longer tube tends to disappear [13]. Based on the physiological mechanism of tube formation, a mathematical model is proposed [14]: Suppose that the pressures at nodes i and j are p_i and p_j, respectively, and that the two nodes are connected by a cylinder of length L_{ij} and radius of the cylinder is r_{ij}. According to the $Hagen - Poiseuille$ equation in chemistry, the flux through the tube can be described as $Q_{ij} = \frac{\pi r^4 (p_i - p_j)}{8\eta L_{ij}} = \frac{D_{ij}(p_i - p_j)}{L_{ij}}$ where η is the viscosity of the fluid, and $D_{ij} = \frac{\pi r^4}{8\eta}$ is a measure of the conductivity of the tube. As the length L_{ij} is a constant, the behavior of the network is described by the conductivity D_{ij}.

The foraging principle of *Physarum* has been used to solve the maze-problem. The two special nodes corresponding to the food sources are named N_1 and N_2 and other nodes are designated N_3, N_4, N_5, and so forth.

N_1 always acts as a source and N_2 as a sink. We assume zero capacity at each node, by considering the conservation law of sol we have $\sum_i Q_{ij} = 0 \quad \text{if} \quad j \neq 1, 2.$

The source node N_1 and the sink node N_2 meet $\sum_i Q_{i1} + I_0 = 0$ and $\sum_i Q_{i2} - I_0 = 0$, respectively. Where I_0 represents the flux of the source node (or into the sink node). It should be noted that I_0 is a constant in our model, which means that the total flux is fixed constant throughout the process.

Experimental observations show that tubes with larger flux expand, while those with smaller flux shrink gradually. Such adaptation behaviours of plasmodium can be described according to the conductivity D_{ij} changing over time with the flux Q_{ij}. More specifically, the evolution of conductivity D_{ij} can be described as follows:

$$\frac{dD_{ij}}{dt} = f(|Q_{ij}|) - rD_{ij} \tag{1}$$

Based on the mathematical model, a network model has been proposed and a lot of experiments have been done. Different parameters are discussed in different situations. This model was also used to find the shortest path between designated cities through freeways in United States. Experiments show that the *Physarum* model can not only find the shortest path successfully but also has efficient reconfiguration ability in the case of traffic accidents.

2.2 A Brief Introduction to Internetware

Web services are web-based, self-described, modular and distributed computing model. Its main idea is to describe software as services while Internetware belongs to this paradigm. As the function of individual services is quite limited, and can not meet the requirements of different users at most of the time, individual services can be combined to provide more powerful composite services. Dynamic composition or reconfiguration is necessary as an atomic service consisting of a composite service may fails during running time.

Internetware entities consist of different components, which can be published in open environments such as the Internet. Those entities can be connected across networks. The development processes of Internetware systems are different from traditional software systems due to the open and dynamic characteristics of the network environments. More specifically, Internetware systems are required to perceive the changes of the external network environments, and then adapt accordingly in regard to the functionality, performance and trustworthness. The development activities of Internetware systems are treated as combining basic software resources without orders into a basic system in order. This process is an iterative one as the system in order may become out-of-order with time goes by [15] [16].

Different Web services published on the Internet can be viewed as different components for Internetware. The service composition process is to combine services without order to ordered ones to form a new basic system.

3 Bio-inspired Service Composition and Reconfiguration Model

In this section, we present the Physarum inspired service composition and reconfiguration model for Internetware systems. For better understanding the model, a problem description is given first.

3.1 Problem Description

Generally, there are different service composition problems - sequential, circular, parallel, and so forth. As other service composition problems such as circular and parallel service composition problems can be converted to sequential problem

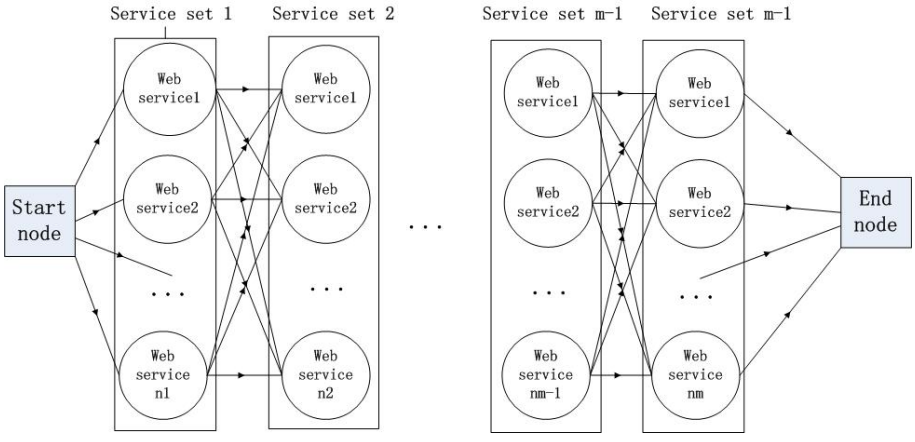


Fig. 1. Relationship among candidate services

by changing some conditions [17], we focus on sequential service composition problem only in this paper.

Suppose there are m different sets of Web services in an Internetware system consisting of Web service components, and there are various service components in each service set.

The relationship among those candidate services can be represented by the directed acyclic graph shown in Figure 1 where n_1 is the number of web service in service set one, n_2 is the number of web service in service set two, and so forth. Therefore, the total number of the combination services is $n_1 \times n_2 \times \cdots \times n_m$. With a relatively large number of services, the service composition problem can not be solved by traditional exhaustive methods. Here, we turn to *Physarum* and propose a *Physarum*-like algorithm to solve this problem.

The formation of this ordered system is similar to *Physarum* maze-solving problem. The users and registry can be viewed as the start and end points in the maze. Different services are viewed as different nodes in the maze. The relation among services is corresponding to the flow among nodes of the maze. Therefore, the mechanism of *Physarum* protoplasm flow changes to find the shortest path can be used to solve the service composition problem – a problem to find the optimal service combination according to the change of relations of the internal components for an Internetware system. This problem is essentially reduced to the problem seeking an optimal path in a weighted directed acyclic graph. As *Physarum* has the ability to find the shortest path in an intricate maze according to the length of paths from the start to the end, we can follow it to find the optimal combination of services from the service nodes with different QoS (Quantity of Service).

3.2 Mathematical Model

Usually, services composition performance is defined based on its QoS. Five generic QoS criteria for service composition are proposed in [18], which are service execution cost, service execution duration, satisfaction degree of service, reliability, and availability. In this paper, service execution cost, service execution duration and reliability are chosen as our quality criteria, which are the weights of the relation edges.

According to the *Physarum* maze-solving model, some parameters of web service combination are introduced into our model. Suppose that the pressures at nodes i and j in adjacent service sets are p_i and p_j, and the service execution costs are $cost_i$ and $cost_j$, respectively. The total execution cost could be the sum of these two nodes when the relation is established between these two nodes, which is $C_{ij} = cost_i + cost_j$. Similarly, the total execution duration could be $T_{ij} = time_i + time_j$, where $time_i$ and $time_j$ are the execution duration at nodes i and j, respectively . However, the reliability K_{ij} is defined as $K_{ij} = K_i \times K_j$, where K_i and K_j are the reliability at nodes i and j, respectively. According to the definition above, the relation strength RS is defined as follows:

$$RS_{ij} = \frac{D_{ij}(p_i - p_j)K_{ij}}{C_{ij} \times T_{ij}} = D_{ij}(p_i - p_j) \times QoS \tag{2}$$

The change behavior of network is described by the conductivity D_{ij} of every relation edge. The QoS of relations among services are measured by $K_{ij}/C_{ij} \times T_{ij}$. For any node i, its reliability is defined as $K_i = \frac{req_i}{req_{sum}}$.

We define users and registry as N_1 and N_2. For any intermediate service nodes i and j, based on the Kirchhoff's law, we can obtain $\sum_i RS_{ij} = 0 \quad \text{if} \quad j \neq 1, 2$. The equations to describe nodes N_1 and N_2 are $\sum_i RS_{i1} + I_0 = 0$ and $\sum_i RS_{i2} - I_0 = 0$, respectively, where I_0 is a constant.

To simulate the adaptive behaviors of service components, the conductivity of service components evolves as described in $\frac{dD_{ij}}{dt} = f(|RS_{ij}|) - D_{ij}$.

The first term in the right side of the equation indicates that the conductivity increases with increasing of the relation strength, while the second term denotes the shrinking rate of the relation edges. In the case of low relation strength, relations will gradually disappear.

The service relationship between the components can be expressed by a seven-tuple, denoted as $R(ID_1, ID_2, WSL_1, WSL_2, WSD_1, WSD_2, RS)$, where ID_1 and ID_2 are unique identifiers of two related services; WSL_1 and WSL_2 are URLs of the two services; WSD_1 and WSD_2 are service descriptions,; and RS represents the relation strength between services.

3.3 Algorithm Characteristics

The characteristics of the service composition and reconfiguration algorithms described above can be summarized as follows:

- This slime mold amoeba - *Physarum polycephalum* can establish a reliable adaptive network efficiently in the foraging process. A world class team in Japan and the United Kingdom has been conducting research on this for quite a long time [11] – [14] and [19] – [24], which laid a solid theoretical foundation for such approaches.
- The proposed service combination and reconfiguration algorithms integrate both Web service and and amoeba based features.
  item Our approach is based on global optimization, which will ensure the optimal combination and reconfiguration with correct parameters.
- Our reconfiguration method can be obtained through real-time index, the iterative calculation that can select the best QoS service for the user through its own parameters automatically evolution with non-human intervention and non-stop.

4 Experimental Results and Discussion

In our experiments, the execution price, execution duration and reliability of the node connections are all randomly generated in the experiments in the range of $[0, 1]$. The number of services sets is 10, with start node and end node, a total of 12 categories of services, the number of web services in each service set is 10 which are present by the following web service composition and reconfiguration simulation.

Assume that what is needed is a combination of 10 different services; each type of service has 10 service nodes provide the same service. Only two services in adjacent service sets could have edges. There is no relation edge between non-adjacent set of services and services within the same service set. Then the process of finding the optimal combination is finding the of strongest relation edge from the beginning to the end nodes.

Services Composition Simulation Experiments: In the bitmap, each node represents a service that service composition starts at the left-most single node in a column and ends at the right-most single node in a column. The nodes listed between the start and end nodes in the same column are services to provide the same functionality, the connection line between two points, represent the two nodes (service) relation. The thickness of Line represents the relation strength. The initial state is shown in Figure 2.

When the simulation experiment begins, the strength of relationship will gradually changes with repeated iterations. Ultimately, it will choose some node of different type from the beginning to the end for the optimal path, that is, the optimal combination of services.

Reconfiguration Simulation Experiments: The process of reconfiguration simulation experiment is as follows:

Simulation results based on a combination of the above, assuming that relations edge of$e453$ and $e546$ failure, service node 64 is not available, the system will select a new optimal services through its own parameters automatically evolution with non-human intervention and non-stop as shown in Figure 3 (top).

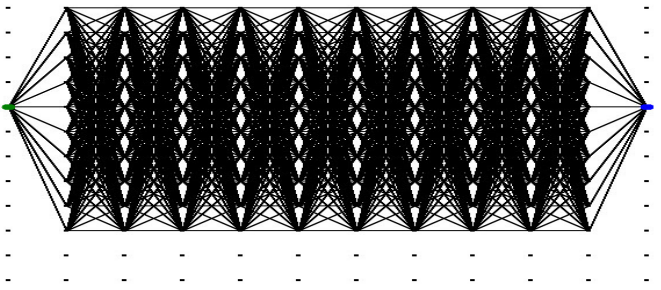


Fig. 2. Initial state

The system automatically selects the other sub-optimal node to ensure that the software system automatically is reconfigured (Figure 3, bottom).

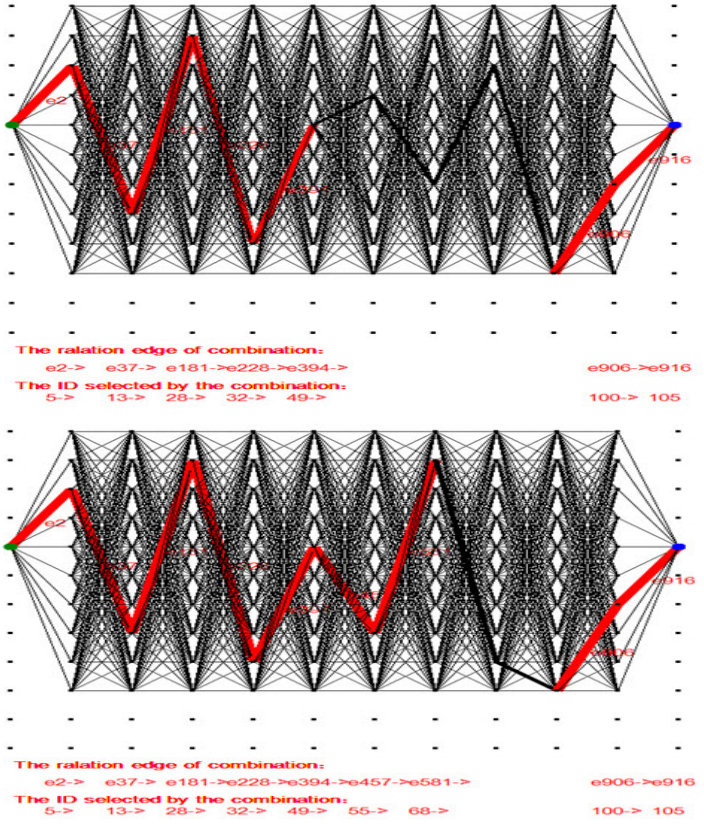


Fig. 3. The process of reconfiguration(b)

The final reconfiguration result is shown in Figure 4.

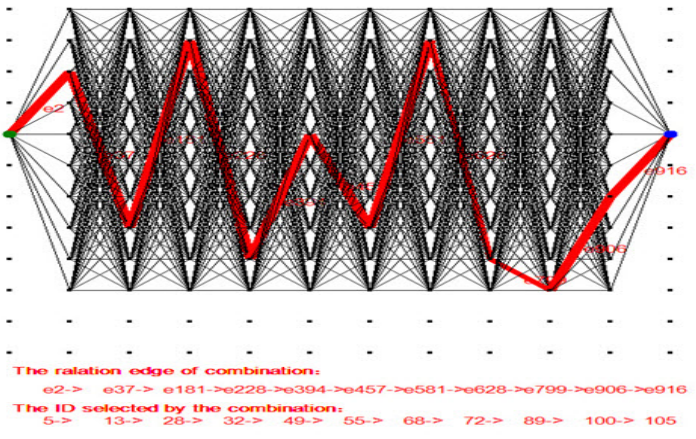


Fig. 4. Result of reconfiguration

5 Conclusions

We focused on solving the problem of service-oriented Internetware component composition and dynamic reconfiguration. In response to this kind of problem, there are a number of related researches. This paper is based on the *Physarum polycephalum* maze-solving bionic model. It can be seen from the experiment, our method can find the optimal combination of services by QoS, and if some changes happened in the environment, with non-stop and non-human intervention the adaptive reconfiguration succeed in finding replacement nodes to achieve the best combination to ensure excellent. Our approach provides a new way for dynamic web service composition and reconfiguration; some parameters in the model will be adjusted and optimized in our future work.

References

1. Ni, W.-c., Liu, L.-c., Wu, C.: Survey on Web Services Composition Methods. Computer Engineering 04, 79–81 (2008) (in Chinese)
2. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.-C.: Adaptive and Dynamic Service Composition in eFlow. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 13–31. Springer, Heidelberg (2000)
3. Hamadi, R., Benatallah, B.: A Petri net-based model for web service composition. Australian Computer Society, Inc., Adelaide (2003)
4. Liao, J., Tan, H., Liu, J.-D.: Describing and Verifying Web Science Using Pi-Calculus. Chinese Journal of Computers 28(04), 635–643 (2005) (in Chinese)
5. Li, M., Wang, D.-Z., Du, X.-Y., Wang, S.: Dynamic Composition of Web Services Based on Domain Ontology. Chinese Journal of Computers 28(04), 644–650 (2005) (in Chinese)

6. Ribeiro-Justo, G.R., Saleh, A., Karran, T.: Intelligent Reconfiguration of Dynamic Distributed Components. Electronic Notes in Theoretical Computer Science 180(2), 91–106 (2007)
7. Anosike, A.I., Zhang, D.Z.: An agent-based approach for integrating manufacturing operations. International Journal of Production Economics 121(02), 333–352 (2009)
8. Reece, S., Rogers, A., Roberts, S., et al.: Rumours and reputation: evaluating multi-dimensional trust within a decentralised reputation system. In: Proceedings of 6th AAMAS, Hawaii, pp. 1–8 (2007)
9. Benatallah, B., Dumas, M., Sheng, Q.Z., et al.: Declarative composition and peer-to-peer provisioning of dynamic Web services. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, pp. 297–308 (2002)
10. Nol, L.B., Bellissard, L., De Palma, N., et al.: Dynamic Reconfiguration of Agent-Based Applications. In: Third European Research Seminar on Advances in Distributed Systems (ERSADS apos 1999), Madeira, Island, pp. 23–28 (1999)
11. Tero, A., Takagi, S., Saigusa, T., et al.: Rules for Biologically Inspired Adaptive Network Design. Science 327(5964), 439–442 (2010)
12. Nakagaki, T., Yamada, H., Toth, A.: Intelligence: Maze-solving by an amoeboid organism. Nature 407(6803), 470 (2000)
13. Tero, A., Kobayashi, R., Nakagaki, T.: Physarum solver: A biologically inspired method of road-network navigation. Physica A: Statistical Mechanics and its Applications Information and Material Flows in Complex Networks 363(01), 115–119 (2006)
14. Tero, A., Kobayashi, R., Nakagaki, T.: A mathematical model for adaptive transport network in path finding by true slime mold. Journal of Theoretical Biology 244(04), 553–564 (2007)
15. Lu, J., Ma, X.-X., Tao, X.-P., Xu, F., Hu, H.: Internetware Survey. Science in China (Series E) 36(10), 1037–1080 (2006) (in Chinese)
16. Yang, F.-Q., Mei, H., Lu, J., Jin, Z.: Some Discussion on the Development of Software Technology. Acta Electronica Sinica 30(12A), 1902–1906 (2002) (in Chinese)
17. Zeng, L., Benatallah, B., Ngu, A.H.H., et al.: QoS-aware middleware for Web services composition. IEEE Transactions on Software Enineering 30(05), 311–327 (2004)
18. Zeng, L., Benatallah, B., Dumas, M., et al.: Quality driven web services composition. In: Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, pp. 411–421 (2003)
19. Nakagaki, T., Yamada, H., Ueda, T.: Interaction between cell shape and contraction pattern in the Physarum plasmodium. Biophysical Chemistry 84(03), 195–204 (2000)
20. Nakagaki, T., Yamada, H., Hara, M.: Smart network solutions in an amoeboid organism. Biophysical Chemistry 107(01), 1–5 (2004)
21. Nakagaki, T., Kobayashi, R., Nishiura, Y., et al.: Obtaining multiple separate food sources: behavioural intelligence in the Physarum plasmodium. Proceedings of the Royal Society of London, B: Biological Sciences 271(1554), 2305–2310 (2004)
22. Tero, A., Yumiki, K., Kobayashi, R., et al.: Flow-network adaptation in Physarum amoebae. Theory in Biosciences 127, 89–94 (2008)
23. Nakagaki, T., Iima, M., Ueda, T., et al.: Minimum-Risk Path Finding by an Adaptive Amoebal Network. Phys. Rev. Lett. 99(6), 068104-1–068104-4 (2007)
24. Marwan, W.: Amoeba-Inspired Network Design. Science 327(5964), 419–420 (2010)

A Novel Search Interval Forecasting Optimization Algorithm

Yang Lou[1], Junli Li[1], Yuhui Shi[2], and Linpeng Jin[1]

[1] Information Science and Engineering College, Ningbo University, 315211 Ningbo, China
[2] Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, 215123 Suzhou, China
awl0015@126.com, li.junli@vip.163.com, yuhui.shi@xjtlu.edu.cn, kinis1984@163.com

Abstract. In this paper, we propose a novel search interval forecasting (SIF) optimization algorithm for global numerical optimization. In the SIF algorithm, the information accumulated in the previous iteration of the evolution is utilized to forecast area where better optimization value can be located with the highest probability for the next searching operation. Five types of searching strategies are designed to accommodate different situations, which are determined by the history information. A suit of benchmark functions are used to test the SIF algorithm. The simulation results illustrate the good performance of SIF, especially for solving large scale optimization problems.

Keywords: Global Numerical Optimization, Evolutionary Algorithm, Search Interval Forecasting.

1 Introduction

Global numerical optimization is an important research topic because many real-world problems can be described as global numerical optimization problems. Many numerical optimization problems are difficult to solve and some of them even can not be solved by analytical methods. Consequently, Evolutionary Algorithms (EAs) [1], which simulate the natural processes of evolution, were proposed to solve these problems. In CEC (IEEE Congress on Evolutionary Computation) 2005, a special competition of real-parameter optimization was hold and 17 algorithms [2] were proposed, all of which had been tested on a suite of 25 benchmark functions with 10 and 30 dimensions, respectively [3]. Furthermore, another special competition on solving large scale global optimization problems using EAs [4] was organized in CEC 2008. The test suite was designed based on 7 benchmark functions with 100, 500 and 1000 dimensions, respectively [4]. Eight algorithms were selected to enter the final competition, among which MTS (Multiple Trajectory Search) [5] proposed by Tseng and Chen won the first prize. MTS uses multiple agents to search the solution space concurrently [6]. Nevertheless, other algorithms also contributed a lot to improve global numerical optimization methods as well. Yang et.al proposed the MLCC (Multilevel Cooperative Coevolution) [7], in which an improved framework of

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 374–381, 2011.
© Springer-Verlag Berlin Heidelberg 2011

cooperative coevolution was designed to overcome the hard-to-determine parameters. In this paper, we proposed a novel search interval forecasting (SIF) optimization algorithm, which utilizes the history information accumulated in the previous iteration of the evolution. The framework of search interval forecasting algorithm is designed for solving large scale optimization, but it has downward compatibility for solving lower dimensional problems.

The remainder of this paper is organized as follows. Section 2 gives some definitions of optimization and subinterval used in this paper. Section 3 introduces and describes the SIF algorithm. Experimental simulation and result discussion are given in Section 4, followed by conclusions in Section 5.

2 Preliminaries

2.1 Problem Definition

Without loss of generality; in this paper, we consider minimization problems without constraint. The global numerical optimization problem is defined as follows:

$$\text{Minimize } f(\boldsymbol{x}), \text{ subject to } \underline{\boldsymbol{x}} \le \boldsymbol{x} \le \overline{\boldsymbol{x}}$$

where point $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)$ is a variable vector in the search space R^N, and $f(\cdot)$ is a single-objective function without constraint. Vectors $\underline{\boldsymbol{x}} = (\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_N)$ and $\overline{\boldsymbol{x}} = (\overline{x}_1, \overline{x}_2, \cdots, \overline{x}_N)$ define the lower and upper bound of the variable vector $\boldsymbol{x}$, i.e. each dimension x_i of a feasible solution must satisfy $\underline{x}_i \le x_i \le \overline{x}_i, i = 1, 2, ..., N$. The feasible solution space is denoted by $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$. The fitness value of a potential solution (a point in the search space) of the minimization problem is defined as $-f(\boldsymbol{x})$.

2.2 Subinterval Definition

The search range of each dimension of the variable vector $\boldsymbol{x}$ in the search space is denoted by $[\underline{x}_i, \overline{x}_i]$, $i = 1, 2, ..., N$. The interval $[\underline{x}_i, \overline{x}_i]$ is divided into M subintervals.

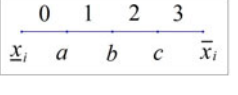


Fig. 1. One dimension search range segmented into 4 equal subintervals

As shown in Fig 1, the search range is segmented into 4 ($M = 4$) equally sized subintervals, thus $b = 0.5(\underline{x}_i + \overline{x}_i)$, $a = 0.5(\underline{x}_i + b)$ and $c = 0.5(b + \overline{x}_i)$. The numbers in the top are the indexes of the subintervals. Through trial-and-error method, we found $M = 4$ is a good value. When $M = 4$, the indexes range from 0 to 3. We name subinterval $[\underline{x}_i, a]$ as *No.*0subinterval , $[a, b]$ as *No.*1subinterval , $[b, c]$ as

*No.*2subinterval and $[c, \bar{x}_i]$ as *No.*3subinterval . Search Interval Forecasting (SIF) algorithm utilizes the history information of these subintervals accumulated in the previous iteration to forecast the next searching subinterval, but ignores those subintervals that have low probability to find the global optimal according to the history information, therefore SIF can reduce the objective function evaluations.

3 Search Interval Forecasting Optimization Algorithm

3.1 Parameters Definition

The index of the current searching subinterval is represented by *iCur* . SIF marks three points in each subinterval, i.e. the current best point X_b , which has the current best fitness value of all and its index is denoted by iX_b ; the best point before the current best (pre-best) point pX_b , and the best points before the pre-best point ppX_b . A Boolean value *yes* / *no Cur* denotes whether the searching subinterval is feasible, which is determined by the history information.

We define "succeed" of a searching operation as "selecting a set of points randomly in the specific subinterval and its fitness value is larger than the current best". Otherwise, it is a failure trial of a searching operation. Five types of trial times of searching operations are kept. They are succeed times *Sts* , failure times *Fts* , the total trial times *Tts* , the succeed times of searching operations in the "+" direction Sts^+ , and the succeed times of searching operations in the "−" direction Sts^- .

SIF algorithm has two queues to record the points which have specific attributes, i.e. the smooth queue *SQ* and the failure queue *FQ* . The points in the smooth queue *SQ* have the same fitness value as that of the current best point. If the current best point changes, and the smooth queue will be updated. Consequently, *SQ* contains three types of points, "the best point" X_b , pX_b and ppX_b described before.

In the failure queue *FQ* , each record includes three items of information: the position of current point *Xpt* , the residual values δ_f of objective function between the current point and the current best point, and a Boolean value denotes whether a further local search is needed. The minimum δ_f is marked as $\min \delta_f$, which is the minimum residual of objective function values between the current point and the current best point in the failure queue *FQ* .

3.2 Five Types of Searching Strategies

We define five types of strategies for the searching in subintervals, the nearest + direction searching strategy(NS^+), and the nearest − direction searching strategy (NS^-), the longest distance searching strategy(*LS*), the minimum operable residual + direction searching strategy(MRS^+), and the minimum operable residual − direction searching strategy(MRS^-). Among these strategies, a searching operation

is defined as "selecting a set of points randomly in the specific subinterval". $NS^{+/-}$ search strategies mean searching in the next + or – direction subinterval. *LS* means searching in the subinterval which has a longest distance to the current point. And $MRS^{+/-}$ strategies mean searching in the subinterval which has the minimum residual with the current subinterval.

3.3 SIF Summarization

The SIF algorithm shown in Fig.2 can be summarized as follows:

Step 1. It is the Initialization step. For each dimension, a midpoint of any subinterval of 4 subintervals is selected randomly, and then a vector of the initial point is constructed. Record the five types of trial times of searching Sts , Fts , Tts , Sts^+ , and Sts^- ; and record information of X_b , pX_b , ppX_b , iX_b , $iCur$ and $\min \delta_f$ as well.

Step 2. For each dimension, five types of searching strategies are utilized.
If the current subinterval is the same as the current best subinterval i.e. $iCur = iX_b$, NS^+ and NS^- are utilized. If $Sts^+ > Sts^-$, NS^+ is utilized first, otherwise, NS^- first. If the searching is successful, ignore the rest of **Step 2** and go to **Step 3**;

If $iCur \neq iX_b$, the rest three types of searching strategies are utilized.

Searching strategy *LS* is utilized, and if the searching is successful, ignore the rest of **Step 2** and go to **Step 3**; if the three strategies above are unsuccessful, then MRS^+ and MRS^- are utilized.

Step 3. If there is a successful searching in **Step 2**, then update all parameters and history information.

Otherwise, there is no successful searching, a value of subinterval is calculated as

$$No.j\text{subinterval}.value = No.j\text{subinterval}.\min \delta_f + \frac{No.j\text{subinterval}.Fts}{No.j\text{subinterval}.Sts}$$

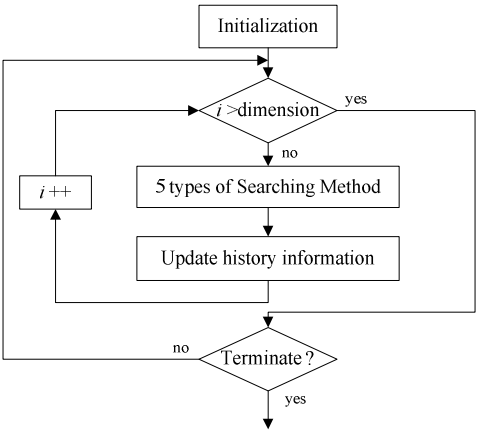


Fig. 2. Flow Chart of SIF

where *No.j*subinterval ($j = 1,2,3,4$) means a subinterval, *Sts* , *Fts* and $\min \delta_f$ are defined above. For four subintervals, the subinterval with the minimum *No.j*subinterval.*value* is treated to be the next searching subinterval.

Step 4. If number of objective function evaluations reaches its pre-fixed maximum value, the algorithm stops, otherwise, go to **Step 2**.

4 Experimental Studies

4.1 Benchmark Functions

Six benchmark functions are adopted here. The dimensions of each function are set to be 100, 500 and 1000, respectively. A fixed number of Function Evaluations (*FEs*) is given for each problem. The performance of an algorithm is quantitatively measured by the value of objective functions, and the *FEs* is defined as

$$FEs = 5000 \times \text{Dimensionality}$$

And for each problem, 25 independent runs are carried out. Among the series of testing, we focus on the problems with dimension 1000. Table.1 shows the six benchmark functions in our experiment.

Table 1. Benchmark Functions

Benchmark Functions	Search Range	Optimal Solution Value
1. Sphere Function $f_1(\boldsymbol{x}) = \sum_{i=1}^{D} \boldsymbol{x}_i$	$[-100,100]^D$	$\text{Min } f_1 = f_1(0,0,\cdots,0) = 0$
2. Schwefel's Problem 2.21 $f_2(\boldsymbol{x}) = \max_i \{\lvert \boldsymbol{x}_i \rvert, 1 \le i \le D\}$	$[-100,100]^D$	$\text{Min } f_2 = f_2(0,0,\cdots,0) = 0$
3. Rosenbrock's Function $f_3(\boldsymbol{x}) = \sum_{i=1}^{D-1} (100(\boldsymbol{x}_i^2 - \boldsymbol{x}_{i+1})^2 + (\boldsymbol{x}_i - 1)^2)$	$[-100,100]^D$	$\text{Min } f_3 = f_3(0,0,\cdots,0) = 0$
4. Rastrigin's Function $f_4(\boldsymbol{x}) = \sum_{i=1}^{D} (\boldsymbol{x}_i - 10\cos(2\pi \boldsymbol{x}_i) + 10)$	$[-5,5]^D$	$\text{Min } f_4 = f_4(0,0,\cdots,0) = 0$
5. Griewank Function $f_5(\boldsymbol{x}) = \frac{1}{4000} \sum_{i=1}^{D} \boldsymbol{x}_i^2 - \prod_{i=1}^{D} \cos(\frac{\boldsymbol{x}_i}{\sqrt{i}}) + 1$	$[-600,600]^D$	$\text{Min } f_5 = f_5(0,0,\cdots,0) = 0$
6. Ackley's Function $f_6(\boldsymbol{x}) = 20 - 20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} \boldsymbol{x}_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi \boldsymbol{x}_i))$	$[-32,32]^D$	$\text{Min } f_6 = f_6(0,0,\cdots,0) = 0$

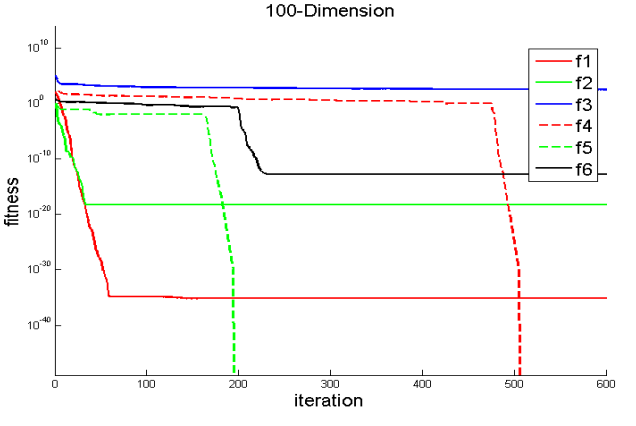
4.2 Experimental Data

Table 2 shows the average results of 25 independent run for each problem. Six benchmark functions with 100, 500 and 1000 dimensions are tested within 5×10^5, 2.5×10^6 and 5×10^6 times of *FEs*, respectively. For function f_4, SIF can get the optimum in all 25 runs. As we know, the higher dimension a problem has, the more difficult it is to optimize. This can be inferred from simulation results. For function f_5, from Table 2, SIF can solve it with 100 dimensions accurately, and can obtain results with the precision of 10^{-15} for the problem with 500 and 1000 dimensions.

In Fig.3, processes of convergence are displayed, in which the Y-axis is set in the form of log and the minimum displayed is less than 10^{-40}. For functions f_1, f_2, f_5 and f_6, SIF has fast convergence speed, while for f_3, the convergence process is relatively longer because of f_3 is a hard-to-solve problem, which is multi-modal and non-separable. For function f_4, SIF can solve it within 600 iterations.

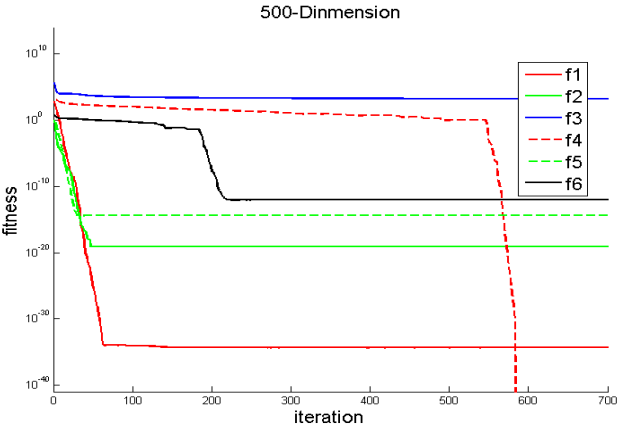
Table 2. The Average Results of 25 Independent Runs

Functions	f_1	f_2	f_3	f_4	f_5	f_6
100-Dimension (5×10^5 *FEs*)	6.4706E-36	5.5849E-19	2.8877E+2	0.0000E+0	0.0000E+0	1.7452E-13
500-Dimension (2.5×10^6 *FEs*)	4.3108E-35	7.1876E-20	1.4893E+3	0.0000E+0	1.1102E-15	8.4598E-13
1000-Dimension (5×10^6 *FEs*)	8.5512E-35	3.5223E-19	2.8601E+3	0.0000E+0	3.9968E-15	1.6417E-12

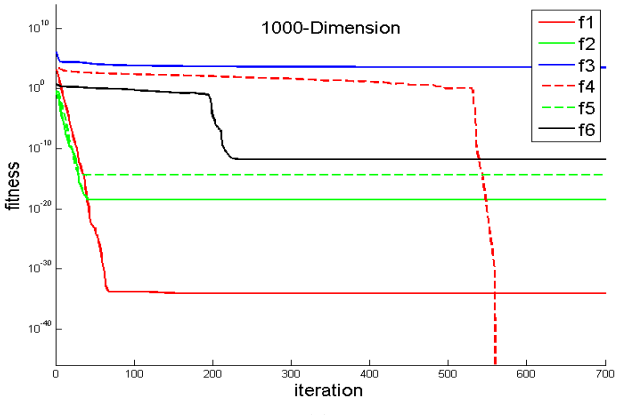


(a)

Fig. 3. Convergence Process Graphs for Optimizations with (a) 100-Dimension; (b) 500-Dimension and (c) 1000-Dimension Problems



(b)



(c)

Fig. 3. (*Continued*)

5 Conclusion

In this paper, a novel population-based optimization algorithm, search interval forecasting algorithm, is proposed. A suit of benchmark functions are used to test the validity of the proposed SIF algorithm. Even though the experimental study is very preliminary but it do illustrate the effectiveness of the proposed SIF algorithm. More experimental studies, algorithm analysis, and comparison with other existing algorithms are our future work.

Acknowledgment

This research was supported by the Natural Science Foundation of China under Grant Nos.60832003, 60975080, the Natural Science Foundation of Zhejiang Province

under Grant No.Y1100076, the Natural Science Foundation of Ningbo under Grant No.2009A610089 and K.C. Wong Magna Fund in Ningbo University.

References

1. Engelbrecht, A.P.: Computational Intelligence: An Introduction, 2nd edn. Wiley Publishing. Inc., Chichester (2009)
2. http://www3.ntu.edu.sg/home/epnsugan/
3. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC, Special Session on Real-Parameter Optimization. Technical Report, Nanyang Technological University, Singapore, and KanGAL Report #2005005, IIT Kanpur, India (2005)
4. Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: Benchmark Functions for the CEC'2010 Special Session and Competition on Large Scale Global Optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, & Nanyang Technological University (2009)
5. Tseng, L.Y., Chen, C.: Multiple Trajectory Search for Large Scale Global Optimization. In: Proc. of the IEEE Congress on Evolutionary Computation, pp. 3052–3059 (2008)
6. Tseng, L.Y., Chen, C.: Multiple Trajectory Search for Multiobjective optimization. In: Proc. of the IEEE Congress on Evolutionary Computation, pp. 3609–3616 (2007)
7. Yang, Z.Y., Tang, K., Yao, X.: Multilevel Cooperative Coevolution for Large Scale Optimization. In: Proc. of the IEEE Congress on Evolutionary Computation, pp. 1663–1670 (2008)

A Danger Theory Inspired Learning Model and Its Application to Spam Detection

Yuanchun Zhu and Ying Tan

Key Laboratory of Machine Perception (MOE), Peking University
Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China
{ychzhu,ytan}@pku.edu.cn

Abstract. This paper proposes a Danger Theory (DT) based learning (DTL) model for combining classifiers. Mimicking the mechanism of DT, three main components of the DTL model, namely signal I, danger signal and danger zone, are well designed and implemented so as to define an immune based interaction between two grounding classifiers of the model. In addition, a self-trigger process is added to solve conflictions between the two grounding classifiers. The proposed DTL model is expected to present a more accuracy learning method by combining classifiers in a way inspired from DT. To illustrate the application prospects of the DTL model, we apply it to a typical learning problem — e-mail classification, and investigate its performance on four benchmark corpora using 10-fold cross validation. It is shown that the proposed DTL model can effectively promote the performance of the grounding classifiers.

Keywords: artificial immune system, danger theory, machine learning, spam detection.

1 Introduction

The development of Artificial Immune System (AIS) is usually promoted by the proposal of novel Biological Immune System (BIS) paradigms. In recent years, a novel biological paradigm — Danger theory (DT), proposed by Matzinger [1], has become popular in explaining the mechanism of BIS. According to the DT, an immune response is not triggered by the detection of 'non-self' but the discovery of 'danger', and immunity is controlled by an interaction between tissues and the cells of the immune system. Although there are still debates on the relation between the DT and classical viewpoint, the DT does contain enough inspiration for building relative AIS [2]. Based on DT, novel AIS paradigms have been proposed and applied to web mining and intrusion detection. Secker et al. [3] presented a DT based adaptive mailbox, where high number of unread messages were defined as the source of danger. Aickelin et al. [4] gave thoughts about the way of building a next generation Intrusion Detection System (IDS) based on DT. In Ref. [5], the development and application of two DT based algorithms for intrusion detection, namely the Dendritic Cell Algorithm and the Toll-like Receptor Algorithm, were presented.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 382–389, 2011.
© Springer-Verlag Berlin Heidelberg 2011

In this paper, we propose a DT based learning (DTL) model for combining classifiers. Mimicking the mechanism of DT, signal I, danger signal and danger zone are designed for machine learning task, and then the framework of the model is presented. Among the three components, danger zone is the most important one leading to the success of the DTL model. The danger zone defines a specific way of interaction between two grounding classifiers. To illustrate the application prospects of the DTL model, we apply it to a typical classification task — spam detection, and investigate its performance on four benchmark corpora using 10-fold cross validation. Experiments were conducted to analyze the effect of the danger zone, and compare the DTL model with classical machine learning approaches. It is shown that the proposed model can effectively promote the performance of the grounding classifiers.

The remainder of the paper is organized as follows. In Section II, we present how to transplant the three main concepts of DT into the machine learning task, and the framework of the DTL model is given. In Section III, the DTL model is implemented for spam filtering task. Section IV discusses our experimental results. The conclusions are presented in Section V.

2 Danger Theory Based Learning Model

The immune system has the ability of detecting and responding to dangerous things, according to DT. This phenomenon implies that the immune system can discriminate between danger and non-danger. Thus, it is logical to build a DT based Learning model for two-group classification problem. In this section, we concern with how to transplant the three main concepts of DT, namely Match — Signal I, Danger Signal and Danger Zone, into the field of machine learning.

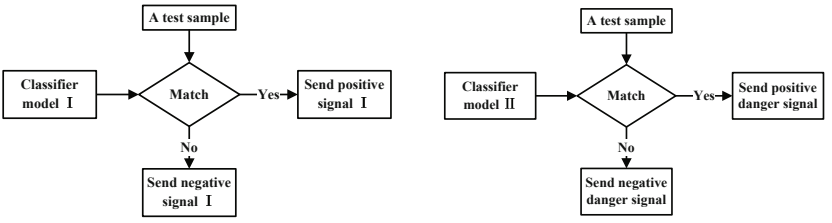
2.1 Generating Signals

The signal I is generated using the classifier I for each test sample in the DTL model. The process is depicted in Fig. 1(a). When the classifier I classify a test sample as positive class (match occurs), it will send a positive signal I to the sample. Otherwise, it will send negative one to the sample, if no match occurs.

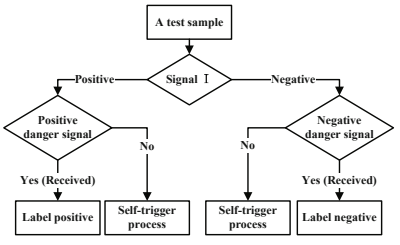
Figure 1(b) shows how a danger signal (Signal II) is triggered by the classifier II. Although the generating process of a danger signal seems to be quite similar as that of a signal I, the transmission coverage of a danger signal is quite different from that of a signal I. When a signal I is triggered, it will be sent only to the specific sample, upon which the signal is arisen. However, a triggered danger signal will be sent to all the test samples within the danger zone, besides the specific sample.

2.2 Classification Using Signals

This phase is the key procedure of the DTL model, which defines an immune based interaction between the two classifiers. As shown in Fig. 1(c), a test sample



(a) The process of generating signal I (b) The process of generating signal II



(c) Classification process using signals

Fig. 1. The process of classification using signals

is labeled only if the two signals that it received agree with each other. Otherwise, a self-trigger process is utilized to get the test sample classified.

The weighted result given by the interaction between the two classifiers is defined as Eq. 1.

$$E(x_i) = \sum_{x_j \in D} \delta(c_1(x_i), c_2(x_j)) K(d(x_i, x_j)), \tag{1}$$

where x_i and x_j are test samples, D denotes the test set, $c_1(x)$ and $c_2(x)$ are the two classifier models, $d(x_i, x_j) = \parallel x_i - x_j \parallel$ is the distance between two samples, $K(z)$ is defined in Eq. 2, and $\delta(y_1, y_2) = 1$, if $y_1 = y_2$, and 0 otherwise.

$K(z)$ defines the effect of the danger zone as follows.

$$K(z) = \begin{cases} 1 & if\ z \leqslant \theta \\ 0 & otherwise \end{cases}, \tag{2}$$

where θ is the size of the danger zone.

After obtaining the weighted result $E(x_i)$, the sample x_i can get its class label using Eq. 3.

$$L(x_i) = \begin{cases} c_1(x_i) & if\ E(x_i) \geqslant 1 \\ f(x_i) & otherwise \end{cases}, \tag{3}$$

where $f(x)$ denotes the class label given by the self-trigger process.

Algorithm 1. Framework of the DTL model

```
Select two uncorrelated classifiers: classifier I, II;
Train the two classifiers respectively on train corpus;
for each sample x_i in test corpus do
  Trigger a signal I upon x_i using classifier I and send the signal to x_i;
  Trigger a danger signal upon x_i using classifier II and send the signal to the test
  samples within the danger zone of x_i;
end for
for each sample x_i in test corpus do
  if x_i has received a positive signal I then
    if x_i has received a positive danger signal then
      Label x_i as positive class;
    else
      Call self-trigger process;
    end if
  else
    if x_i has received a negative danger signal then
      Label x_i as negative class;
    else
      Call self-trigger process;
    end if
  end if
end for
```

Self-Trigger Process: for the test samples which get conflict results from classifier I and II, a self-trigger process is designed. An intuitional thought is to get the sample activated using its nearest neighbor. Thus, the Nearest Neighbor (NN) approach is applied in this phase [6]. In future work, we intend to incorporate other approaches for self-trigger process into the DTL model and compare their performance.

2.3 The Framework of the DTL Model

Algorithm 1 summarizes the framework of the DTL model, in which two grounding classifiers interact through two signals. In the model, two grounding classifiers are first chosen and trained independently. Then the signal I and the danger signal, simulating the signals in the DT, are triggered upon each test sample utilizing the two classifiers. Finally, each test sample gets labeled by considering the interaction between the two classifiers.

2.4 Analysis of the DTL Model

For any machine leaning model, the essence of it is the conditional probability $P(y_k|x_i)$ of class y_k that it computes for each test sample x_i. In the DTL model, a test sample x_i gets a label y_k in two cases as follows.

(1)The two grounding classifiers give a consistent label y_k to the sample x_i: Suppose the two grounding classifiers are conditionally independent, given a test

sample x_i. Then the probability $P(y_k \mid x_i, case_1)$, which denotes the probability that the two grounding classifiers give consistent label y_k to the sample x_i, is computed as follows.

$$\begin{aligned} &P(y_k \mid x_i, case_1) \\ \leqslant &P(c_1(x_i) = y_k \mid x_i) \cdot \sum_{x_j \in D} P(c_2(x_j) = y_k \cap K(\parallel x_i - x_j \parallel) = 1 \mid x_i, x_j). \end{aligned} \tag{4}$$

(2)There is confliction between the two grounding classifiers, and the self-trigger process gives the label y_k to the sample x_i. The probability $P(y_k \mid x_i, case_2)$, which denotes the probability that this case may happen, is defined as follows.

$$P(y_k \mid x_i, case_2) = P(E(x_i) = 0 \cap f(x_i) = y_k \mid x_i). \tag{5}$$

Following the above analysis, the probability $P(y_k \mid x_i)$, computed by the DTL model, is presented in Eq. 6.

$$P(y_k|x_i) = P(y_k \mid x_i, case_1) + P(y_k \mid x_i, case_2), \tag{6}$$

which can be expanded using Eqs. 4 and 5.

3 Filter Spam Using the DTL Model

3.1 Feature Extraction

At the beginning, terms are selected according to their importance for classification, which can be measured by Information Gain (IG) [7].

Bag-of-Words (BoW), also referred to as vector space model, is usually utilized as the feature extraction approach for spam filtering [8]. In BoW, an email is transformed into a d-dimensional vector $\langle x_1, x_2, \ldots, x_d \rangle$ by calculating occurrence of previously selected terms. For BoW with Boolean attribute, x_i is assigned to 1 if t_i occurs in the e-mail, or it is assigned to 0 otherwise. In our experiments, 300 features were selected by using IG, and a BoW with Boolean attribute was applied to the feature extraction phase.

3.2 Selection of Classifiers

Support Vector Machine (SVM) and Naive Bayes (NB) are chosen as the two grounding classifiers of the DTL model, as they are two of the most prevalent and effective classifiers especially for spam filtering [9,10].

3.3 Performance Measures

To validate the effectiveness of the proposed DTL model, two overall performance measures were adopted in our experiments, namely accuracy and F_β measure [8]. The two components of F_β measure are also given in the experiment results.

4 Experiments of Spam Detection

Experiments were conducted on four benchmark corpora PU1, PU2, PU3, and PUA[1] [11], using 10-fold cross validation. The corpora have been preprocessed when published, by removing attachments, HTML tags, and header fields except for the subject. The details of the corpora can be found in Ref. [11].

4.1 The Effects of the Danger Zone

The specific interaction between the two grounding classifiers is implemented by the design of the danger zone. To some extent, the success of the DTL model lies in a proper danger zone design and an optimal size of the danger zone. In this subsection, we investigate the impact of the danger zone size on the overall performance of the DTL model. Experiments of the DTL model with different danger zone size were conducted on PU1, using 10-fold cross validation. The results are depicted in Fig. 2(a), which shows the variational performance of the DTL model, as the size of the danger zone growing larger. At initial stages, the accuracy and F_1 measure increases as the size of the danger zone is enlarged. Then, the performance of the DTL model peaks at a size of 20. After that, the performance declines as the size growing even larger.

4.2 Comparison Experiments

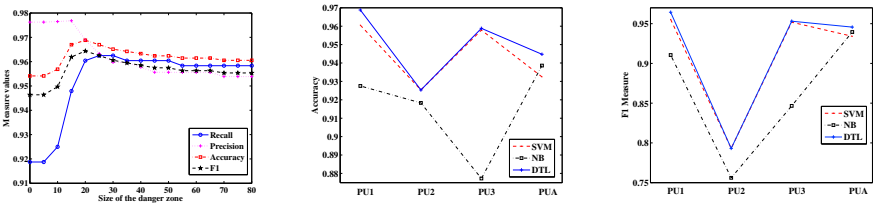
Comparison experiments were conducted on four benchmark corpora PU1, PU2, PU3, and PUA to validate the proposed DTL model, using 10-fold cross validation. As the four corpora have already been preprocessed when published, our experiments began at the phase of feature extraction. First, 300 discriminative words were selected by using the IG method. Then based on this, each e-mail was transformed to a 300-dimensional feature vector. Finally, the two grounding classifiers were built from the feature vector set.

In the experiments, two performance measures — accuray and F_1 measure were adopted as mentioned in Section 3.3. Fig. 2(b) depicts the comparison of accuracy among SVM, NB and DTL, while Fig.2(c) shows the comparison of F_1 measure among the three approaches. More details on the comparison are shown in Table 1, where the two components of F_1 measure, namely spam recall and spam precision, are also given. Besides, the performance of NN, which was utilized in self-trigger process, is also presented in the table.

On corpus PU1, PU3 and PUA, the DTL model outperforms SVM and NB in terms of both accuracy and F_1 measure. On corpus PU2, the DTL model performs equally as SVM and outperforms NB. From these results, we can draw a preliminary conclusion that the proposed DTL model can effectively improve the performance of classifiers.

Why does the DTL model perform not so outstandingly on corpus PU2 as it does on the three other corpora? The preliminary investigation shows that the

[1] The four PU corpora can be downloaded from the web site: http://www.aueb.gr/users/ion/publications.html



(a) Performance of the DTL model with different danger zone size

(b) Accuracy of SVM, NB and DTL on corpus PU1, PU2, PU3 and PUA

(c) F_1 measure of SVM, NB and DTL on corpus PU1, PU2, PU3 and PUA

Fig. 2. Performance of SVM, NB and DTL on corpus PU1, PU2, PU3 and PUA

Table 1. Performance of SVM, NB, NN and DTL on four PU corpora

Corpus	Approach	**Recall**	**Precision**	**Accuracy**	F_1	**Feature dim.**
PU1	SVM	95.83%	95.39%	96.06%	95.54%	300
	NB	85.00%	98.30%	92.75%	91.06%	300
	NN	84.17%	94.43%	90.73%	88.86%	300
	DTL	96.04%	96.89%	**96.88%**	**96.44%**	300
PU2	SVM	72.86%	88.72%	**92.54%**	**79.31%**	300
	NB	65.71%	91.00%	91.83%	75.60%	300
	NN	45.71%	84.13%	87.32%	58.52%	300
	DTL	72.86%	88.72%	**92.54%**	**79.31%**	300
PU3	SVM	94.45%	96.04%	95.79%	95.19%	300
	NB	77.25%	94.03%	87.72%	84.66%	300
	NN	84.51%	95.38%	91.33%	89.57%	300
	DTL	94.73%	95.99%	**95.88%**	**95.31%**	300
PUA	SVM	94.56%	92.60%	93.25%	93.41%	300
	NB	94.39%	93.98%	93.86%	93.95%	300
	NN	90.88%	86.87%	87.98%	88.39%	300
	DTL	95.44%	93.93%	**94.47%**	**94.57%**	300

two grounding classifiers make more correlated error on corpus PU2 compared to other corpus. This reflects that the success of the DTL model lies in selection of uncorrelated grounding classifiers. Besides, the poor performance of the self-trigger process (NN) on PU2 is also a reason for the unideal performance of the DTL model.

5 Conclusions

In this paper, we have transplanted the main concepts of the DT into building an immune based learning model. In addition, the DTL model has been successfully applied to a typical machine learning problem – spam detection. The experimental results show that the proposed DTL model can promote the performance of

grounding classifiers. In the experiments, the DTL model outperformed SVM, NB and NN in terms of both accuracy and F_1 measure.

In future work, we seek to incorporate other design of danger zone and self-trigger process into the DTL model, and investigate the performance of the model under different settings. In this way, we hope to obtain a more ideal model and better performance. Finally, we intend to add other signals, which can indicate the drift of knowledge, into the DTL model. In this way, we hope it can develop into an adaptive learning model.

Acknowlegements. This work was supported by National Natural Science Foundation of China (NSFC), under Grant No. 60875080 and 60673020, and partially supported by the National High Technology Research and Development Program of China (863 Program), with Grant No. 2007AA01Z453.

References

1. Matzinger, P.: The danger model: a renewed sense of self. Science's STKE 296(5566), 301–305 (2002)
2. Aickelin, U., Cayzer, S.: The danger theory and its application to artificial immune systems. In: Proceedings of the First International Conference on Artificial Immune Systems, pp. 141–148. Citeseer (2002)
3. Secker, A., Freitas, A., Timmis, J.: A danger theory inspired approach to web mining. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 156–167. Springer, Heidelberg (2003)
4. Aickelin, U., Bentley, P., Cayzer, S., Kim, J., McLeod, J.: Danger Theory: The Link between AIS and IDS? In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 147–155. Springer, Heidelberg (2003)
5. Aickelin, U., Greensmith, J.: Sensing danger: Innate immunology for intrusion detection. Information Security Technical Report 12(4), 218–227 (2007)
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
7. Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In: Proceedings of International Conference on Machine Learning, pp. 412–420. Citeseer (1997)
8. Guzella, T., Caminhas, W.: A review of machine learning approaches to Spam filtering. Expert Systems with Applications 36(7), 10206–10222 (2009)
9. Drucker, H., Wu, D., Vapnik, V.: Support vector machines for spam categorization. IEEE Transactions on Neural networks 10(5), 1048–1054 (1999)
10. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A Bayesian approach to filtering junk e-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, vol. 62, pp. 98–105. AAAI Technical Report WS-98-05, Madison (1998)
11. Androutsopoulos, I., Paliouras, G., Michelakis, E.: Learning to filter unsolicited commercial e-mail. Technical report, National Centre for Scientific Research "Demokritos", Greece (2006)

Research of Hybrid Biogeography Based Optimization and Clonal Selection Algorithm for Numerical Optimization

Zheng Qu and Hongwei Mo

Automation College, Harbin Engineering University, Harbin, 150001, China

Abstract. The interest of hybridizing different nature inspired algorithms has been growing in recent years. As a relatively new algorithm in this field, Biogeography Based Optimization(BBO) shows great potential in solving numerical optimization problems and some practical problems like TSP. In this paper, we proposed an algorithm which combines Biogeography Based Optimization (BBO) and Clonal Selection Algorithm (BBOCSA). Several benchmark functions are used for comparison among the hybrid and other nature inspired algorithms (BBO, CSA, PSO and GA). Simulation results show that clone selection can enhance the ability of exploration of BBO and the proposed hybrid algorithm has better performance than the other algorithms on some benchmarks.

Keywords: Biogeography based optimization, Clonal selection algorithm, Optimization.

1 Introduction

Nature inspired algorithms, such as Genetic Algorithm, Particle Swarm Optimization, Clonal Selection Algorithm, and Biogeography Based Optimization, have been proposed for solving optimization problems in the last few decades. The characteristics of these algorithms are different for different problems and there is no the best algorithm for all problems. So many researchers had been trying to hybridize different algorithms for more effective applications.

Biogeography researches the migration of species from one isolated habitat to another, the increase and distinction of species. A good habitat has a high habitat suitability index (HSI), which is related to the factors such as rainfall, diversity of vegetation, diversity of topographic feature, land area and temperature. Suitability index variables (SIVs) are those variables characterizing habitability. A large number of species tend to be in habitats with a high HIS, while a small number in habitats with a low HSI. Species in habitats with a high HSI are more likely to emigrate to nearby habitats which means high emigration rate, simply driven by the large number of species that they host. At the same time, habitats with a high HSI have a low immigration rate because of the saturation state they already have. Therefore, the species distribution in high HSI habitats is more stable than that in low HIS habitats.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 390–399, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Inspired by the theory, in 2008, Simon proposed Biogeography Based Optimization (BBO) inspired by concepts from biogeography[1]. BBO is mainly composed of the migration operator and the mutation operator based on immigration rate and migration rate. By the above two operators poor solutions will share some new characters from good solutions to raise the quality of those solutions. Like EAs and swarm intelligence, BBO is also a population-based optimization algorithm. However, it has unique features of maintaining the set of solution and only some local variables of solution change during the optimization process. BBO has been applied to many optimization problems. In the past two years, many improvements to BBO had been finished by Simon and some other researchers[2][3][4][5][6][7]. But there are few researches about hybrid BBO with the other algorithms. Aniruddha et al [4] presented a hybrid technique combining differential evolution with biogeography – based optimization (DE/BBO) algorithm to solve both convex and nonconvex economic load dispatch (ELD) problems. In [8], Gong et al investigated the possible combination between BBO and DE for numeric optimizations. An attempt to hybridizing BBO and PSO for Cross-Country Path Finding was discussed in [9]. A hybrid of flower pollination by artificial bees (FPAB) and BBO was propsed in [10] for Satellite Image Classification. Bhattacharya et al propsed hybrid BBO and DE for solving economic load dispatch problem[11].

In the field of Artificial Immune Systems(AIS),the clonal selection theory has been used as inspiration for the development of AIS that perform computational optimization and pattern recognition tasks. de Castro and Von Zuben [12] developed one of the most popular and widely used clonal selection inspired AIS called CLONALG, which has been used to performed the tasks of pattern matching and multi-modal function optimisation. Like negative selection, clonal selection has proven to be very popular in the AIS community spawnng a great deal of research with recent examples including [13][14][15].For the CSA, Carlos A. Coello et al [16] demonstrated an approach to hybridize Genetic Algorithm and Artificial Immune System for global optimization. In [17], Wang et al proposed a hybrid Particle Swarm Optimization (PSO) method, which is based on the fusion of the PSO, Clonal Selection Algorithm (CSA), and Mind Evolutionary Computation (MEC).

We proposed a hybrid algorithm of BBO and CSA. Since the performance of the original BBO is not very good at some optimization problems, it needs to be improved in further. We use the ability of local search and diversity production of CSA to improve the optimizing ability of BBO.

This paper is organized as follows. After the introduction, section 2 illustrates a brief review of BBO and CSA. Section 3 demonstrates the framework of the hybrid and pseudo-code of implementation. Section 4 reports the performance of BBOCSA on several benchmark functions in the literature, comparing with that of BBO, Clonal Selection Algorithm, PSO and GA.

2 BBOCSA

2.1 Framework of BBOCSA

For balancing the exploration and exploitation of the algorithm, we hybridize Biogeography Base Optimization and Clonal Selection Algorithm because BBO has

shown considerable speed of convergence while CSA has great ability in local searching.

The process of the BBOCSA is as follows.

Initializing: In this step, set up parameters of BBOCSA and generate a population of candidate solutions.

Evaluating fitness: Calculate fitness value, corresponding to each candidate solution. At the same time, any infeasible solutions are replaced with a random feasible solution.

Selection: From all solutions in the current population, the best solutions are selected for the next clone step, the number of which is a constant among the parameters in Initializing step.

Clone: The selected solutions have different clonal amounts due to different fitness. The better a selected solution is, the larger its clonal amount is.

Hypermutation: After the clone process, all candidate solutions have to go through hypermutation with a possibility Pm. Gaussian mutation is used here.

Emigration and immigration operator: First of all, according to the fitness values, every candidate solution has a corresponding species number. Then an emigration rate and an immigration rate are obtained by employing a migration model.

Termination Criterion: Terminate after some generations. Otherwise, it goes back to the step Evaluating fitness.

2.2 The Procedure of BBOCSA

The parameters used in the algorithm are as follows.

Algorithm 1. The main procedure of BBOCSA

```
Generate the initial population P
Evaluate the fitness for each individual in P
While the termination criterion is not satisfied Do
      Select the best nselect individuals in P to
      form new population
      Obtain new population by cloning the selected
      individuals according to clone operator
      Modify current population with the
      hypermutation operator
      Apply migration operator to the current
      population
End While
```

Algorithm 2. Clone operator of BBOCSA

```
For i = 1 : PopSize Do
      CloneCount = CloneMax + 1 – i
```

```
 3:          For j = 1 : CloneCount Do
 4:                To duplicate one more same individual
                   as the i_th individual in P
 5:          End For
 6:    End For
```

Algorithm 3. Hypermutation operator of BBOCSA

```
 1:    For i = 1 : PopSize Do
 2:          MutateRate = rand
 3:          If rand < MutateRate Then
 4:                For j = 1 : dimensionNum Do
 5:                      If rand < MutateRate Then
 6:                        Scale = rand * rand *
                          (MaxParValue - MinParValue)
 7:                        P_i.chrom_j = P_i.chrom_j + + Scale *
                           randn;
 8:                      End If
 9:                End For
10:          End If
11:    End For
```

Algorithm 4. Migration operator of BBOCSA

```
 1:    For i = 1 : PopSize Do
 2:          For each individual, map the fitness to the
             number of species
 3:          Calculate the immigration rate λ_i and the
             emigration rate μ_i for each individual P_i
 4:    End For
 5:    For i = 1 : PopSize Do
 6:          Normalize the immigration rate lambdaScale
             for P_i in the range of 0 and 1
 7:          For j = 1 : dimensionNum Do
 8:                If rand < lambdaScale Then
 9:                      Select an individual P_select to
                         emigrate by roulette
10:                      P_i.chrom_j = P_i.chrom_j + rand *
                         rand * (P_Select.chrom_j - P_k.chrom_j)
11:                End If
12:          End For
```

The main process of BBOCSA is shown in Algorithm 1. Algorithm 2 and 3 give the clone operator and hypermuation operator, which are the key strategy of CSA. Algorithm 4 shows the migration operator of BBOCSA. In this step, the cloned habitats will have many copies of itself. They have high probability to mutate to be individuals with high HSIs in hypermuation process. So in the migration process,

more habitats with high HSIs will share good information with those poor ones. While in the original BBO, the habitats with high HSI have no copies of themselves. The diversity of habitats is lower than that of BBOCSA. In fact, the introduction of clone strategy enhances the ability of global search and the diversity generation for BBO. In [9], Ma lists six migration models for Biogeography Based Optimization and compares the performance of the different models for solving 23 benchmark functions. We apply linear migration model for this paper.

3 Experimental Results

For a clear observation and a fair judgment on the performance of BBOCSA, we employed 7 widely used benchmark functions of numerous optimizations and compared BBOCSA with BBO, CSA, PSO and GA. All of benchmark problems are 30 dimensions. Each benchmark function is run30 times and gets the statistics results.

BBO is the original one without mutation operator introduced by [8].PSO is the also the one used in [8], whose inertial constant is 0.3 and whose cognitive constant, social constant for swarm interaction and social constant for neighborhood interaction are all set to 1. GA employs a single point crossover operator and a single point mutation operator. Crossover probability is 1 and mutation probability 0.01. The test results are shown in Fig.1.

From Figure.1 A to G, we can see that BBOCSA shows faster convergence than BBO and CSA except Quartic_Noise. PSO and GA cannot compete with BBOCSA either on the convergence rate or the optimal solutions they found. Table1 shows the comparison results of BBOCSA with the other algorithms. It competes with the other algorithms and has relative better results on most of the test problmes. For most benchmark functions BBOCSA could not guarantee an optimal solution after termination, which means a better local searching scheme should be introduced to BBO in the future.

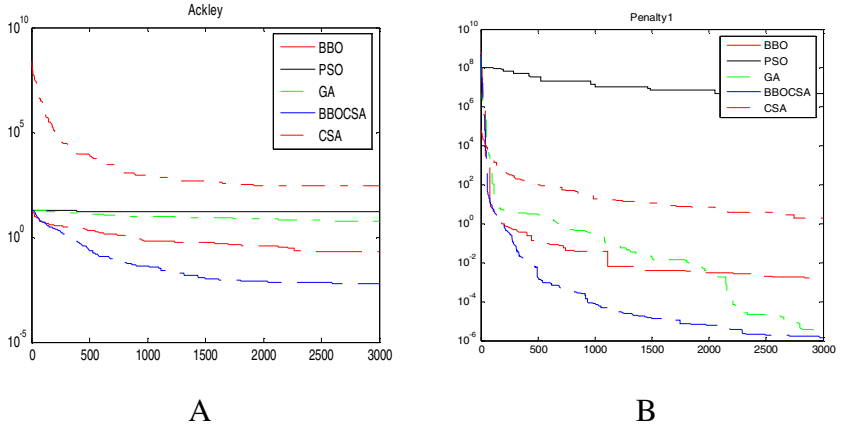
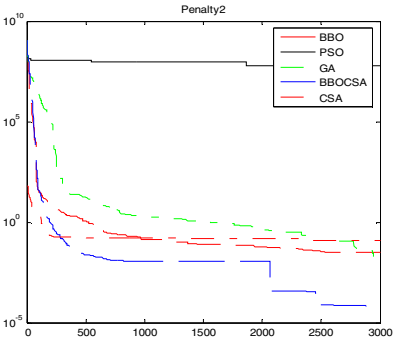
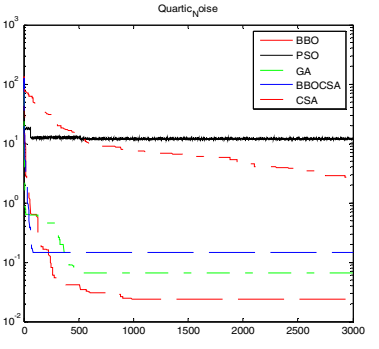


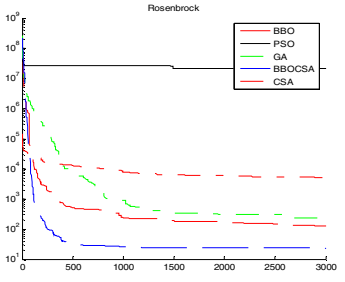
Fig. 1. The comparison of convergence of BBOCSA with the other optimization algorithms



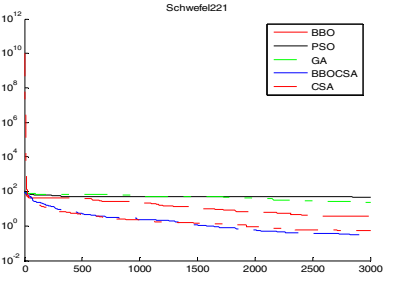
C



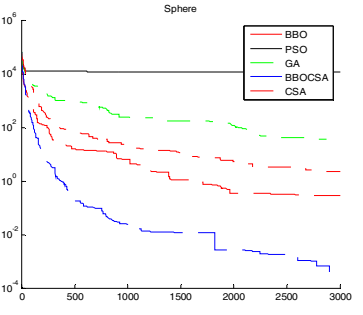
D



E



F



G

Fig. 1. (*Continued*)

Table1. The comparison BBOCSA and the other algorithms

		Best	Mean	std	median	worst
BBO CSA	Sphere	6.73E-05	2.55E-04	1.56E-04	2.31E-04	7.73E-04
	Schwefel221	1.96E-03	5.13E-03	1.46E-03	4.89E-03	8.29E-03
	Rosenbrock	3.18E+01	1.05E+02	4.36E+01	9.57E+01	2.08E+02
	Quartic Noise	1.88E-01	3.16E-01	7.44E-02	3.14E-01	4.84E-01
	Ackley	2.04E+00	6.91E+01	3.82E+01	7.95E+01	1.71E+02
	Penalty1	0	0	0	0	0
	Penalty2	5.23E-02	1.78E-01	6.16E-02	1.80E-01	3.33E-01
CSA	Sphere	1.359336	2.2575552	0.5850292	2.1383805	3.553987
	Schwefel221	4.83E-01	5.99E-01	7.86E-02	5.87E-01	7.60E-01
	Rosenbrock	1.80E+03	3.65E+03	1.33E+03	3.53E+03	7.11E+03
	Quartic Noise	2.19E+00	2.96E+00	3.78E-01	2.95E+00	3.96E+00
	Ackley	6.98E+01	2.34E+02	7.94E+01	2.44E+02	4.55E+02
	Penalty1	0.00E+00	2.17E+00	1.37E+00	2.00E+00	5.00E+00
	Penalty2	3.11E-02	1.14E-01	3.95E-02	1.18E-01	1.96E-01

Table1. (*Continued*)

BBO	Sphere	0	0	0	0	0
	Schwefel221	9.00E+00	1.79E+01	6.11E+00	1.80E+01	3.10E+01
	Rosenbrock	1.92E+01	3.61E+01	2.28E+01	2.67E+01	8.15E+01
	Quartic Noise	1.28E-06	2.29E-04	2.50E-04	1.49E-04	1.08E-03
	Ackley	1.05E-01	1.99E-01	5.24E-02	1.94E-01	3.34E-01
	Penalty1	1.57E-32	1.57E-32	5.57E-48	1.57E-32	1.57E-32
	Penalty2	1.35E-32	1.35E-32	5.57E-48	1.35E-32	1.35E-32
PSO	Sphere	21.359096	27.966283	3.1518669	27.404425	35.494248
	Schwefel221	3.08E+01	4.10E+01	3.57E+00	4.11E+01	4.92E+01
	Rosenbrock	4.63E+02	6.79E+02	1.36E+02	6.57E+02	9.42E+02
	Quartic Noise	1.27E+01	2.10E+01	5.16E+00	2.09E+01	3.27E+01
	Ackley	1.42E+01	1.53E+01	5.00E-01	1.54E+01	1.60E+01
	Penalty1	6.83E+05	4.21E+06	2.26E+06	4.79E+06	8.82E+06
	Penalty2	6.41E+06	2.55E+07	1.14E+07	2.54E+07	6.29E+07

Table1. (*Continued*)

GA	Sphere	0	0.0964185	0.0987029	0.0655108	0.3829862
	Schwefel221	1.70E+01	3.63E+01	1.16E+01	3.50E+01	6.70E+01
	Rosenbrock	2.82E+01	5.97E+01	2.60E+01	4.95E+01	1.33E+02
GA	Quartic Noise	6.09E-04	7.39E+00	1.13E+01	3.95E-02	4.25E+01
	Ackley	3.56E+00	8.60E+00	2.45E+00	8.46E+00	1.35E+01
	Penalty1	1.57E-32	1.57E-32	5.57E-48	1.57E-32	1.57E-32
	Penalty2	1.35E-32	8.33E-02	1.56E-01	1.35E-32	7.00E-01

4 Conclusions and Future Work

In this paper, we proposed a new hybrid algorithm which combines the biogeography and immune clone selection. The clone selection mechanism can enhance the ability of exploration of BBO. We test the algorithm by six typical functions. The simulation results show that BBOCSA is an effective hybrid optimization algorithm. We will continue to research BBOCSA in more complex problems, such as multi objective and constrained optimization problems.

Acknowledgement. This paper is supported by National Nature Science Foundation of China(No. 61075113), Excellent Young Teacher Foundation of Heilongjiang (No.1155G18).

References

[1] Simon, D.: Biogeography-based optimization. IEEE Trans. on Evolutionary Computation 12, 702–713 (2008)
[2] Simon, D.: A Probabilistic analysis of a simplified biogeography-based optimization algorithm. Evolutionary Computation, 1–22 (2009)
[3] Simon, D., Ergezer, M., Du, D.W.: Population distributions in biogeography-based optimization algorithms with elitism. In: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, TX, pp. 1017–1022 (2009)

[4] Ergezer, M., Simon, D., Du, D.W.: Oppositional biogeography-based optimization. In: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, TX, pp. 1035–1040 (2009)
[5] Du, D.W., Simon, D., Ergezer, M.: Biogeography-based optimization combined with evolutionary strategy and immigration refusal. In: 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, pp. 1023–1028 (2009)
[6] Rarick, R., Simon, D., Villaseca, F. E., Vyakaranam, B.: Biogeography-based optimization and the solution of the power flow problem. In: IEEE Conference on Systems, Man, and Cybernetics, San Antonio, TX, pp. 1029–1034 (2009)
[7] Gong, W.Y., Cai, Z.H., Ling, C.X., Li, H.: A real-coded biogeography-based optimization with mutation. Applied Mathematics and Computation 216, 2749–2758 (2010)
[8] Gong, W.Y., Cai, Z.H., Ling, C.X.: DE/BBO: A Hybrid Differential Evolution with Biogeography Based Optimization for Global Numerical Optimization. In: Soft Computing - A Fusion of Foundations, Methodologies and Applications (2010)
[9] Johal, N.K., Singh, S., Kundra, H.: Cross - Country Path Finding using Hybrid approach of PSO and BBO. International Journal of Computer Applications 7, 15–19 (2010)
[10] Johal, N.K., Singh, S., Kundra, H.: A hybrid FPAB/BBO Algorithm for Satellite Image Classification. International Journal of Computer Applications 6, 31–36 (2010)
[11] Bhattacharya, A., Chattopadhyay, P.K.: Hybrid Differential Evolution with Biogeography – Based Optimization for Solution of Economic Load Dispatch Power Systems. IEEE Transactions on Issue 25, 1955–1964 (2010)
[12] De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6, 239–251 (2002)
[13] Cutello, V., Nicosia, G.: The clonal selection principle for in silico and in vitrocomputing. In: De Castro, L.N., Von Zuben, F.J. (eds.) Recent Developments in Biologically Inspired Computing. Idea Group Publishing, Hershey (2004)
[14] de Mello Honório, L., da Silva, A.M.L., Barbosa, D.A.: A gradient-based artificial immune system applied to optimal power flow problems. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628, pp. 1–12. Springer, Heidelberg (2007)
[15] May, P., Timmis, J., Mander, K.: Immune and evolutionary approaches to software mutation testing. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628, pp. 336–347. Springer, Heidelberg (2007)
[16] Carlos, A., Coelloa, C., Cortésa, N.C.: Hybridizing a Genetic Algorithm with an Artificial Immune System for Global Optimization. Engineering Optimization 36, 607–634 (2004)
[17] Wang, X., Gao, X.Z., Ovaska, S.J.: A Hybrid Particle Swarm Optimization Method Systems. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC 2006), Taipei, pp. 4151–4157 (2006)

The Hybrid Algorithm of Biogeography Based Optimization and Clone Selection for Sensors Selection of Aircraft

Lifang Xu[1,2], Shouda Jiang[1], and Hongwei Mo[3]

[1] Department of Automatic Test and Control, Harbin Institute of Technology
150001 Harbin, China
[2] Engineering Training Center, Harbin Engineering University
150001 Harbin, China
[3] Automation College, Harbin Engineering University
150001 Harbin, China
mxlfang@163.com,jiangsd@hit.edu.cn,honwei2004@126.com

Abstract. Biogeography-based optimization algorithm(BBO) is a new kind of optimization algorithm based on Biogeography. It mimics the migration strategy of animals to solve the problem of optimization. In this paper, the clone selection strategy is combined with biogeography for solving the problem of sensors selection of aircraft. It is compared with other classical nature inspired algorithms. The comparison results show that BBOCSA is an effective algorithm for optimization problem in practice. It provides a new method for this kind of problem.

Keywords: Biogeography, Biogeography-based optimization, Clone selection, Sensor selection.

1 Introduction

In recent years, we have seen that many algorithms inspired by natural phenomenon or mechanisms, including evolutionary algorithms, artificial immune systems and so on. In this paper, we mainly focus on a new kind of algorithm, which is called biogeography based optimization(BBO), which is inspired by the science of biogeography. It is very interesting in that it mimics the migration process of animals to design method for solving engineering problems, especially optimization.

In the early 1960s, Robert MacArthur and Edward Wilson began working together on mathematical models of biogeography. They were interested in mathematical models for the extinction and migration of species. Since their distinct work, biogeography has become a major area of research[1].Mathematical models of biogeography describe how species migrate from one island to another, how new species arise, and how species become extinct. The term "island" here is used descriptively rather than literally. That is, an island is any habitat that is geographically isolated from other habitats.

Simon presented the first paper on biogeography inspired algorithm for engineering[2]. In his creative work, he merged the burgeoning field of biogeography with engineering in order to see how the two disciplines can be of mutual benefit.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 400–407, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Although the idea of application of biogeography to engineering is similar to those nature inspired algorithms mentioned above, it has completely different mechanisms and process from those ones. Some researches about BBO itself and its applications had been done[3][4][5][6].

In the field of Artificial Immune Systems(AIS),the clonal selection theory has been used as inspiration for the development of AIS that perform computational optimization and pattern recognition tasks. de Castro and Von Zuben [7] developed one of the most popular and widely used clonal selection inspired AIS called CLONALG, which has been used to performed the tasks of pattern matching and multi-modal function optimisation. We proposed a hybrid algorithm of BBO and CSA to solve the problem of aircraft sensor selection.

In this paper, the BBOCSA is used for sensors selection problem of aircraft. The paper is organized as follows. Section 2 reviews the ideas BBOCSA. Section 3 provides some simulation results comparing BBOCSA with other optimization methods. Section 4 presents some concluding remarks and suggestions for further work.

2 The Procedure of BBOCSA

The parameters used in the proposed algorithm BBOCSA are illustrated as follows.

P: the current population
PopSize: the number of individuals in current population;
CloneMax: the number of duplicates the best individual should be cloned;
DimensionNum: the number of dimension in the numerous optimization problem
randn: an operator to generate normally distributed random numbers with mean 0 and standard deviation 1.

Algorithm 1. The main procedure of BBOCSA

```
Generate the initial population P
Evaluate the fitness for each individual in P
While the termination criterion is not satisfied Do
      Select the best n selected individuals in P
      to form new population
      Obtain new population by cloning the selected
      individuals according to clone operator
      Modify current population with the
      hypermutation operator
      Apply migration operator to the current
      population
End While
```

Algorithm 2. Clone operator of BBOCSA

```
For i = 1 : PopSize Do
      CloneCount = CloneMax + 1 - i
```

```
 3:          For j = 1 : CloneCount Do
 4:                To duplicate one more same individual
                   as the i_th individual in P
 5:          End For
 6:    End For
```

Algorithm 3. Hypermutation operator of BBOCSA

```
 1:    For i = 1 : PopSize Do
 2:          MutateRate = rand
 3:          If rand < MutateRate Then
 4:                For j = 1 : dimensionNum Do
 5:                      If rand < MutateRate Then
 6:                        Scale = rand * rand *
                          (MaxParValue - MinParValue)
 7:                        P_i.chrom_j = P_i.chrom_j + + Scale *
                           randn;
 8:                      End If
 9:                End For
10:          End If
11:    End For
```

Algorithm 4. Migration operator of BBOCSA

```
 1:    For i = 1 : PopSize Do
 2:          For each individual, map the fitness to the
             number of species
 3:          Calculate the immigration rate λ_i and the
             emigration rate μ_i for each individual P_i
 4:    End For
 5:    For i = 1 : PopSize Do
 6:          Normalize the immigration rate lambdaScale
             for P_i in the range of 0 and 1
 7:          For j = 1 : dimensionNum Do
 8:                If rand < lambdaScale Then
 9:                      Select an individual P_select to
                         emigrate by roulette
10:                      P_i.chrom_j = P_i.chrom_j + rand *
                         rand * (P_Select.chrom_j - P_k.chrom_j)
11:                End If
12:          End For
```

In geography, geographical areas that are well suited as residences for biological species are said to have a high habitat suitability index (HSI). The variables that characterize habitability are called suitability index variables (SIVs). SIVs can be considered the independent variables of the habitat, and HSI can be considered the dependent variable.

Biogeography is nature's way of distributing species, and is analogous to general problem solutions. A good solution is analogous to an island with a high HSI, and a

poor solution represents an island with a low HSI. High HSI solutions resist change more than low HSI solutions. By the same token in BBO, suppose that we have a global optimization problem and a population of candidate individuals. The individual is represented by a *D*-dimensional integer vector. The population consists of $NP = n$ parameter vectors. Each individual is considered as a "habitat" with a habitat suitability index (HSI), which is similar to the fitness of EAs, to measure the individual. A good solution is analogous to an island with a high HSI, and a poor solution indicates an island with a low HSI. High HSI solutions tend to share their features with low HSI solutions. Low HSI solutions accept a lot of new features from high HSI solutions.In TSPBBO, each individual has its own immigration rate λ and emigration rate μ. A good solution has higher μ and lower λ, vice versa. The immigration rate and the emigration rate are functions of the number of species in the habitat. The migration procedure is shown in Algorithm 4. λ and μ can be calculated as Eqn. (1) and (2).

$$\mu_k = \frac{Ek}{n} \tag{1}$$

$$\lambda_k = I\left(1 - \frac{k}{n}\right) \tag{2}$$

where λ_s and μ_s are the immigration and emigration rates when there are S species in the habitat.

3 Benchmark Results

3.1 Sensor Selection of Aircaft

An inlet supplies air to the fan. The air that leaves the fan separates into two streams, one through the engine core, and the other through the bypass duct. The fan is driven by a low-pressure turbine. The air that passes through the engine core goes through a compressor, which is driven by a high-pressure turbine. Fuel is injected and ignited in the combustor to produce hot gas that drives the turbines. The two air streams recombine in the augmentor duct, where additional fuel may be added to increase the temperature. The air leaves the augmentor at a high velocity through the nozzle (which has an adjustable cross section area) and thereby produces thrust.The engine simulation used in this paper is called Modular Aero Propulsion System Simulation (MAPSS) [8], and was written using Matlab Simulink. The controller update rate is 50 Hz. The three state variables used in MAPSS are low-pressure rotor speed, high-pressure rotor speed, and average hot section metal temperature. The discretized time invariant equations that model the turbofan engine can be summarized as

$$\begin{aligned} x(k+1) &= f[x(k), u(k), p(k)] + w_x(k) \\ p(k+1) &= p(k) + w_p(k) \\ y(k) &= g[x(k), u(k), p(k)] + e(k) \ . \end{aligned} \tag{3}$$

where k is the time index, is the three-element state vector,is the three-element control vector, p is the ten-element health parameter vector, and y is the measurement vector. The measurement consists of the outputs of the sensors with which we instrument the engine. The health parameters change slowly over time. Between measurement times their deviations can be approximated by the zero mean noise $w_p(k)$. The noise term $w_x(k)$ represents inaccuracies in the system model, and $e(k)$ represents measurement noise. We can use multiple sensors at a single location if desired. The use of more sensors results in smaller elements Σ, which means that our health estimate will be better. However, there is a point of diminishing returns. The use of more sensors costs more money, and it may not be worth the extra cost to obtain a marginally improved health estimate. The optimality criterion for the health estimation problem can, therefore, be written as

$$J = \sum_{i=1}^{13} \sqrt{\frac{\sum(i,i)}{\sum_0(i,i)}} + \frac{\alpha C}{C_0}. \tag{4}$$

where Σ_0 and C_0 are reference values used for normalization. Σ_0 is the covariance that results if we use all 11 sensors with no duplicates, and C_0 is the financial cost of fitting the aircraft engine with 11 sensors. α is a scale factor that weights the importance of financial cost relative to estimation accuracy. J is the objective function for the health estimation problem. This approach to sensor selection was first proposed using GAs [9]. When BBO is used to solve the problem, J is referred to as the HSI. In general, we want to use a total of N sensors out of K unique sensors (in our example, K =11) with each sensor being used no more than times. (The numerical values of N , K , and M will be problem dependent.) The total number of possible sensor sets is found by the following procedure. First, we generate a polynomial $q(x)$ as

$$\begin{aligned} q(x) &= (1 + x + x^2 + \ldots + x^M)^K \\ &= 1 + q_1 x + q_2 x^2 + \ldots + x^{MK} \end{aligned} \tag{5}$$

The total number of sets containing exactly N sensors is equal to q_N. This is known as the multinomial theorem [10].

3.2 Results and Discussion

In order to explore the benefits of BBOCSA, we compared its performance with five other population-based optimization methods, including Ant Colony Optimization (ACO), Genetic Algorithm(GA), Simple Genetic Algorithm(SGA), Particle Swarm

Optimzation(PSO), Deferential Evolution (DE), Evolution Strategy(ES), Clone Selection Algorithm(CSA). The parameters of BBO are: habitat modification probability=1, maximum immigration and migration rates=1 for each habitat. We did some rough tuning on each of the optimization algorithms to get reasonable performance, but we did not make any special efforts to fine-tune the algorithms. For ACO, we used the following parameters: initial pheromone value $\tau_0 = 1E-5$,pheromone update constant $Q = 20$, exploration constant $q_0 = 1$, global pheromone decay rate $\rho_g = 0.9$, local pheromone decay rate $\rho_t = 0.5$, pheromone sensitivity $\alpha = 2$, and visibility sensitivity $\beta = 6$. For the GA, we used roulette wheel selection, single point crossover with a crossover probability of 0.3, and a mutation probability of 0.1. For PSO, we used initial and ending inertia weight 0.9 and 0.3 respectively, and a social constant 0.7 for swarm interaction. For the ES, we produced offspring $\lambda = 10$ each generation, and standard deviation $\sigma = 1$ for changing solutions. For CSA, we used clone rate of 0.2,mutation rate 0.1. For DE, we used a weighting factor $F = 0.5$ and a crossover constant 0.5.For the SGA, we used single point crossover with a crossover probability of 1, and a mutation probability of 0.01. The clone rate of BBOCSA is 0.2. The mutation rate of BBOCSA is 0.1. Each algorithm had a population size of 50, an elitism parameter of 2, and ran for 500 generations. Figure 1 shows the results of the simulations.

The sensor selection problem can be solved with population-based optimization methods. A population member consists of a vector of integers, with each element in the vector representing a sensor number. The fitness or HSI of a population member is given by (7) . If an invalid sensor set arises during the optimization process due to too many of a certain sensor type, then we replace some of the duplicated sensor types with a randomly chosen sensor to enforce feasibility.

We assumed here that we could use a total of 20 sensors (out of our unique 11 sensors) with each sensor being used no more than four times. The total number of sensor sets to choose from is the coefficient of in the polynomial

$$q(x) = (1 + x + x^2 + x^3 + x^4)^{11} . \tag{6}$$

The coefficient of x^{20} in this polynomial is equal to 3 755 070. That is the total number of sensor sets that must be searched in order to find the minimum value of J in (5). In order to compute J for a single sensor set, we need to solve for for that sensor set. In order to solve for Σ, we need to solve a discrete algebraic Riccati equation (DARE) . This can be done with the DARE function in Matlab's Control SystemToolbox.

Table 1. The comparison results for sensor selection

	ACO	BBO	DE	ES	GA	CSA	PSO	BBOCSA
Mean	8.230	8.025	8.087	8.1492	8.088	8.166	8.193	8.022
Best	8.229	8.021	8.075	8.131	8.078	8.154	8.181	8.015

Table 1 shows the results of the optimization methods on the sensor selection problem.We see that BBOCSA performs the best in terms of both average performance and best performance.

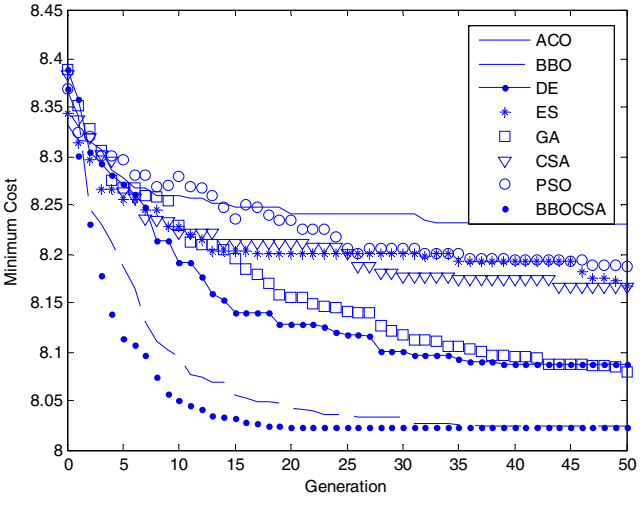


Fig. 1. The comparison results of BBO with the other optimization algorithms

In Fig.1, we can see that BBOCSA and BBO have the relative good performance for the sensor selection problem. And as a new method, BBOCSA shows better performance than the most other nature inspired algorithms, such as ACO, PSO, DE, ES and so on.

4 Conclusion

In this paper, we have combined biogeography and immune clone selection to solve optimization problems in practice. We have applied the BBOCSA to solve the problem of sensors selection of aircraft. The results showed that it provides better performance than the other population-based methods. In future, we will focus on improving its performance on combination optimization problems and solving more complex ones in further.

References

1. Hanski, I., Gilpin, M.: Metapopulation Biology. Academic, New York (1997)
2. Simon, D.: Biogeography-based optimization. IEEE Trans. on Evo. Comp. 12, 702–713 (2008)
3. Simon, D., Ergezer, M., Du, D.W.: Markov Analysis of Biogeography Based Optimization

4. Ergezer, M., Simon, D., Du, D.W.: Oppositional Biogeography-Based Optimization. In: IEEE Conf. on Systems, Man, and Cybernetics, San Antonio, TX, pp. 1035–1040. IEEE Press, New York (2009)
5. Du, D.W., Simon, D., Ergezer, M.: Biogeography-Based Optimization Combined with Evolutionary Strategy and Immigration Refusal. In: IEEE International Conf. on Systems, Man, and Cybernetics, San Antonio, TX, pp. 1023–1028. IEEE Press, New York (2009)
6. Bhattacharya, A., Chattopadhyay, P.K.: Solving Complex Economic Load Dispatch Problems Using Biogeography-based Optimization. Expert Systems with Applications 37, 3605–3615 (2010)
7. De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6, 239–251 (2002)
8. Parker, K., Melcher, K.: The modular aero-propulsion systems simulation (MAPSS) users' guide. NASA, Tech. Memo. 2004-212968 (2004)
9. Mushini, R., Simon, D.: On optimization of sensor selection for aircraft gas turbine engines. In: Proc. Int. Conf. Syst. Eng., Las Vegas, NV, pp. 9–14 (2005)
10. Chuan-Chong, C., Khee-Meng, K.: Principles and Techniques in Combinatorics. World Scientific, Singapore (1992)

A Modified Artificial Immune Network for Feature Extracting

Hong Ge and XueMing Yan

School of Computer Science, South China Normal University, Guangzhou, China
gehong@scnu.edu.cn

Abstract. focusing on "distinctiveness" and "effectiveness" of algorithms inspired by natural immune system, a novel artificial immune network algorithm named immune feature extracting network (IFEN) is proposed to realize the function of feature data extracting in this paper. Based on comprehensive analysis of mechanism of natural immune system and current researching works of artificial immune system (AIS), a modified paradigm of artificial immune network (AIN) and a new mutation operation are designed to adapt to decrease the size of sample data set and extract feature data from the data set with noise. The proposed algorithm is supposed to be used as a data preprocessing method with functions of data compression and data cleansing. Preliminary experiments show that the quality of the data set processed by IFEN is apparently improved and the size is compressed.

Keywords: artificial immune system, artificial immune network, feature extracting.

1 Introduction

Artificial immune system is a general term for different intelligent computing approaches inspired by natural immune system. In the last two decades, a diverse set of immune inspired algorithms to solve various computational problems or tackle real world applications has been produced, and many successful works are reported[1-3,16]. Despite some remarkable works, the field of AIS still lacks unique and effective application domains and algorithm model, which limits the development of AIS to a large degree.

Currently, three major AIS algorithms have been constantly developed and gained popularity[8]: (1) negative selection algorithms (NSA); (2) clonal algorithms (CLONALG); (3) artificial immune networks (AINE). The negative selection algorithm is based upon self/nonself discrimination of T-cells within the thymus, and mainly applied in Computer security, Virus detection etc. Clonal algorithm focused on the clonal selection principle and affinity maturation process of the adaptive immune response and is developed suitable to perform tasks such as machine learning, pattern recognition, and optimization. Artificial immune networks is inspired by immune network theory, which suggests that B-cells are capable of recognizing each other and endow the immune system with a certain type of eigen-behavior and network of communication among cell receptors. AINE is mainly applied in machine learning, pattern recognition.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 408–415, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Clonal selection is the theory used to explain how the immune system responds to nonself antigens. Negative selection in contrast, is one of the strategies used by the immune system to eliminate self-reactive cells, i.e. cells that recognize self antigens. Immune network theory describes how the cells of the immune system interact with other cells of the immune system. Though immune network theory was presented later, it is very interesting and can be used to explain the mechanism of immune system properly and roundly. According to this, artificial immune network can be more reasonable to be an immune inspired approach than clonal selection algorithms. Because of this, our researching work in this paper is focused on inspiration of immune network theory and based on existing artificial immune network models[4-8].

In [9], Garrett tracked the development of AIS, and followed by attempting to make assessment of the usefulness of the AIS in terms of "distinctiveness" and "effectiveness". From this point of view, we attempt to address the function of algorithm inspired by immune system and its application domain by analyzing the functional mechanism of immune system in view of immune network theory. We suggest that feature extraction is one of the most important functions of natural immune system. Starting from this understanding, we propose a modified artificial immune network---immune feature extracting network (IFEN), which is a hybrids of AINE with CLONALG and major function is extracting feature data from a sample data set with noise, and realizes data cleansing and compression.

2 The Immune System: From Information Processing Perspective

The most remarkable roles of immune system are the protection of the organism against the attack of antigen. The primary problem the immune system is faced with, is thus the recognition of these antigen. After recognizing (identifying) an antigen, the immune response arises to avoid or block the antigen, and immune memory which is the feature map of the antigen is memorized in immune system. Therefore, from information processing perspective, the immune system is a highly distributed, adaptive, and self-organizing information processing system, together with its learning, memory, feature extraction, and pattern recognition features, and offers rich metaphors for its artificial counterpart[15]. From the view of biological mechanisms, the main information processing mechanisms of immune system include:

- *Recognition*: when an antigen entering, a set of specific antibodies recognize it based on shape complementary match, and only those antibodies whose match degree, called affinity, with the antigen is over the match threshold will be activated and participate the immune response.
- *Evolution*: those activated antibodies are cloned and mutated, by this way, the affinity of whole population of activated antibodies are improved;
- *Memory*: the antibodies with highest affinity must be preserved somehow as high quality candidate solutions, and shall only be replaced by improved candidates. This immune memory would ensure that both the speed and accuracy of the immune response becomes successively greater after each infection ;
- *Network*: the immune network theory suggests that antibody molecules can be recognized by other antibody molecules. As an outcome of this mutual

recognition of antibody molecules, a network of communication named immune network arises within the immune system and forms a stable memory structure; According to this new perspective of the immune interactions, the interactions of the immune cells are going to result in a network with a natural eigen-behavior whose state will be disturbed by non-self antigens. That is to say, the immune response will cause the component of immune network changed, and in this way, the immune network will be the internal image of the invading antigens;

- *Diversity*: The number of antibodies contained in our immune system is known to be much inferior to the number of possible antigens, making the diversity and individual binding capability the most important properties to be exhibited by the antibody repertoire. The mechanism to maintain diversity of antibody repertoire is specific mutation and concentration suppression of antibodies in immune network.

It is apparent that the most important task of immune system is to recognize, extract and memorize the features of invader antigens. Based on the above mechanisms, natural immune systems can achieve the diversity of antigen recognition and specificity of antigen killing.

3 Immune Feature Extracting Network

Based on the analysis of functional mechanism of immune system and current immune inspired algorithm[10-14], we propose a hybrid of AINE with CLONALG, named Immune Feature Extracting Network (IFEN). The main immune aspects taken into account to develop the algorithm are: selection and cloning of the most stimulated cells proportionally to their antigenic affinity; death of non-stimulated cells; affinity maturation and selection of cells proportionally to their antigenic affinity; and generation and maintenance of diversity.

In our algorithm, antigen refers to data in sample data set, and antibody refers to data in resulting data set handling by IFEN. The IFEN works as follows:

Step 1. Initialization: create an initial random population of network antibodies;
Step 2. Antigenic presentation: for each antigenic pattern, do:

- *Antibodies activation:* for each network element, determine its affinity with the antigen presented. Select a number of antibodies whose affinity with the antigen are more than a pre-specified threshold(activating threshold) as activated set; save the remaining antibodies in a unactivated set.
- *Clonal expansion:* reproduce (clone) activated antibodies proportionally to their affinity; eliminate the antibodies whose clones is less than a pre-specified threshold(selecting threshold);
- *Affinity maturation:* select a number of highest affinity antibodies in activated set and place them into a memory set. Mutate each clone of the remaining activated antibodies inversely proportional to affinity;
- *Clonal suppression:* by incorporating memory antibodies with mutated antibodies, reconstruct the memory set, and then determine the concentration of each antibodies; eliminate antibodies whose concentration is more than a pre-specified threshold;

- *Network construction:* incorporate the clonal memory set with unactivated antibody set, reconstruct the antibody population.
- *Network suppression:* eliminate all network antibodies whose activated counting numbers are less than a pre-specified threshold(living threshold);

Step 3. Cycle: repeat steps 2 until a pre-specified number of iterations is reached.
Step 4. Output: the resulting data are output, and afforded to be sample data for further application.

Several key steps of IFEN are described in details as follow:

- *Affinity definition:* the affinity of antigen Ag_i and antibody Ab_j is computed according to Eq. (1).

$$aff_{ij} = 1/(1+\mathrm{d}(Ag_i, Ab_j)) \tag{1}$$

where d(.) denotes the Euclidian distance between Ag_i and Ab_j.

- *Clonal expansion:* an initial weight value w_j is given to each activated antibody, and keeping the total weight value of activated set unchangeable. Then clone each antibody activated antibodies according to Eq.(2) and eliminate those antibodies whose weight value are lowest;

$$\mathrm{w}_j = \mathrm{int}(\ w_0 \times N_{act} \times \frac{aff_{ij}}{\sum_{j=1}^{N_{act}} aff_{ij}}) \tag{2}$$

Where w_0 is the initial weight value of antibody; N_{act} is the total number of activated antibodies.

- *Affinity maturation:* the main function of this step is to mutate the antibodies in order to improve the affinities of these antibody, and the activated antibodies are mutated according to Eq.(3):

$$\mathrm{Ab}_k = Ab_k + \beta(Ag_i - Ab_k) \tag{3}$$

Where β_k is the mutation rate, and $\beta_k = \mathrm{e}^{1-\frac{\mathrm{N}_{act} * aff_{ik}}{N_{act}}}$, it is inversely proportional to antibody's affinity.

- *Clonal suppression:* the concentration of antibody is evaluated according to Eq(4):

$$C_j = \frac{\sum_{k=1}^{N_{act}} S_{jk}}{N_{act}} \tag{4}$$

Where $S_{\mathrm{jk}} = \begin{cases} 1 & Ab_{jk} / \max\{Ab_{jk}\} \geq \eta \\ 0 & Ab_{jk} / \max\{Ab_{jk}\} < \eta \end{cases}$; $\mathrm{Ab}_{\mathrm{jk}}$ is the affinity of Ab_{j} and Ab_{k}, and η is the concentration suppression threshold.

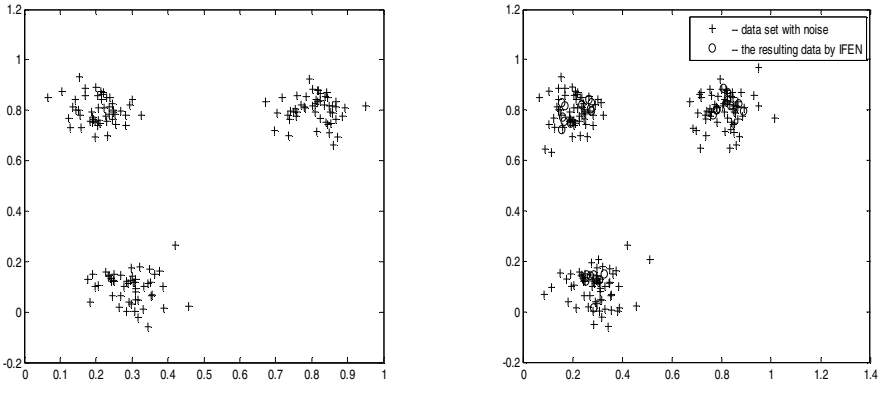
- *Network suppression:* at the beginning of algorithm, the initial activated counting number of each activated antibody is specified to zero. During the iterations, the activated counting number add one whenever the antibody is activated once. If the activated counting number is less than a pre-specified number, the antibody will be eliminated from the antibody population.

4 Experiments Result and Analysis

As mentioned above, the IFEN is used to extract the feature data from a data set, so we will compare the experimental results of using the feature data obtained by IFEN and the initial data set to show the effectiveness of IFEN as a data pre-processing algorithm.

4.1 The Experiments of Clustering Analysis

In this experiment, a data set is generated by program as an initial data set to clustering analysis, and then a number of noise data are added to the initial data set, in this way, a test data set with noise is constructed. The clustering centers of the initial data set and the test data set are respectively determined by K-means method first. Then, processing the test data set by using IFEN, the feature data set of the test data set are obtained by applying the IFEN to the test data, and the clustering centers of the feature data set are obtained by K-means. By comparing the clustering centers of the feature data set and the test data set with the clustering centers of the initial data set, the function of IFEN as a feature data extracting method is verified. The results of the experiments are exhibited in Fig.1.



(a) (b)

Fig. 1. (a) initial data set; (b) data set with noise(i.e. The test data set) and the resulting data set(i.e. the feature data set)

From Fig.1, we can see that the size of the feature data set is apparently decreased by IFEN, the compression rate is nearly 90%. In order to confirm IFEN having function of data cleansing, we compare the clustering centers of the test data set and the feature data set with that of the initial data set, as shown in Table 1. The experiment is repeated for 20 times, and the average errors of clustering centers are calculated shown in last column. The average error of the feature data set is much less than the test data set since the noise is removed by IFEN to a large degree.

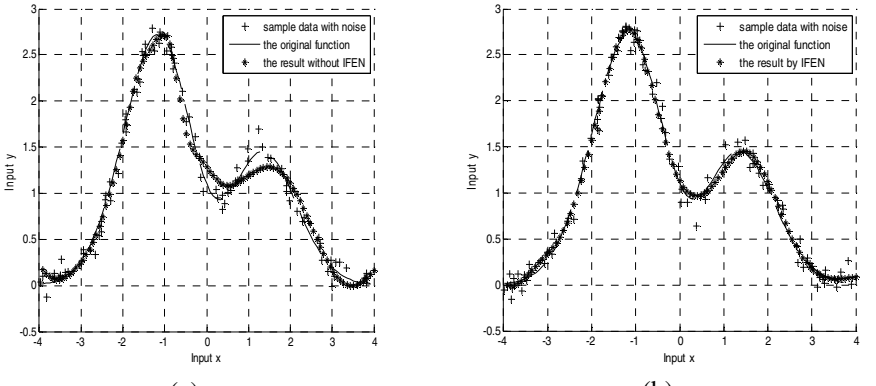
Table 1. The clustering centers of each clustering data set and the average error of clustering centers

clustering centers	initial data set	data set with noise	resulting data set by IFEN
Clustering 1	(0.3296, 0.0456)	(0.2902, 0.1008)	(0.3329, 0.0659)
Clustering 2	(0.5076, 0.8017)	(0.8135, 0.7984)	(0.4814, 0.7969)
Clustering 3	(0.2722, 0.1345)	(0.1956, 0.8078)	(0.2629, 0.1072)
Average error of clustering centers	(0.0000, 0.0000)	(0.1345, 0.1242)	(0.0063, 0.0035)

It is shown that the clustering centers of the feature data set are closer to the clustering centers of the initial data set than that of the test data set. Therefore, IFEN can remove the noise in data set and realize the function of data cleansing and data compression.

4.2 Function Approximation Experiment

In this experiment, randomly selecting sample data from function shown in (5), and add noise to these data, then use these data with noise as sample data to approximate the function. First, without any pre-processing, these sample data are directly used to experiment. Then, handle the data with noise by IFEN, and use the processed data as



(a) (b)

Fig. 2. (a) function approximation with sample data not processed by IFEN; (b) function approximation with sample data processed by IFEN

sample data to experiment. Compare the two results to confirm that IFEN has the function of feature data extracting. The results of the experiments are exhibited in Fig.2.

$$f(x) = 1.1(1 - x + 2x^2)e^{-x^2/2}, \quad x \in [-4,4] \tag{5}$$

In Fig.2(a), the sample data with noise are not processed by IFEN, and directly used to be training data for function approximation. In Fig.2(b), the sample data with noise are processed by IFEN, and then the resulting data are used to be training data. The experimental results show that the approximation errors are using the resulting data processed by IFMN is much less than that of sample data without pre-processing, and the size of sample data is decreased after processing by IFEN. The function of IFEN is verified again.

5 Conclusion and Further Works

In this paper, an attempt is done to address the problems of application domains and modeling of AIS. We have presented a so-called IFEN algorithm based on the analysis of information processing mechanism of natural system and several typical model of AIN. The approach is able to extract feature data from an initial data set with noise and realize data cleansing and data compression. The experimental results indicate that IFMN is effective in practice.

The purpose of this paper is not only to propose a new AIS algorithm, but also to construct a "distinctiveness" and "effectiveness" AIS algorithm, and the latter is more important. The distinctiveness of our algorithm lies in its function of feature data extracting when most existing AIS algorithms are concerned in optimization, pattern recognition and classification etc, so its effectiveness are focused on data preprocessing method to improve the quality of the initial data set. The experiments in this paper are designed to verify the function of IFEN.

However, the experiments in this paper are relatively simple ones. We suggest that IFEN can be a data preprocessing method to be applied in many different areas. More complex experiments will be done concerning different applied domains. Based on analysis of experimental results and theory of immune system, the proposed paradigm will be modified further. Mutation operation is very important for maintaining diversity and ensuring quality of antibody population, and finally effect the performance of the algorithms, so it is worth to do more researching work on this point. We are also interested in how to define the end condition of the algorithms. A general solution is to specify a running times, but it is not the most suitable one. A current weakness of the approach inspired by natural system is lacks of theoretical basis, and there will be a lot of hard and significant works to do in this field[16].

References

[1] Dasgupta, D.: An overview of artificial immune systems. In: Dasgupta, D. (ed.) Artificial Immune Systems and Their Applications, pp. 3–19. Springer, Heidelberg (1998)

[2] Castro, L.N.D., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, London (2002)

[3] Timmis, J.: Artificial immune systems—today and tomorrow. Nat. Comput 6, 1–18 (2007)
[4] Timmis, J.: Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory, University of Wales, Aberystwyth, UK (2000)
[5] Timmis, J., Neal, M., Hunt, J.: An artificial immune system for data analysis. Biosystems 55, 143–150 (2000)
[6] Castro, L.N.d., Zuben, F.J.V.: aiNet: an artificial immune network for data analysis. In: Abbass, H.A., Sarker, R.A., Newton, C.S. (eds.) Data Mining: A Heuristic Approach, pp. 231–259. Idea Group Publishing, USA (2001)
[7] Timmis, J., Neal, M.: A resource limited artificial immune system for data analysis. Knowledge Based System 14, 121–130 (2001)
[8] de Castro, L.N., Timmis, J.I.: Artificial immune systems as a novel soft computing paradigm. Soft Computing 7, 526–544 (2003)
[9] Garrett, S.M.: How do we evaluate artificial immune systems? Evolutionary Computation 13, 145–178 (2005)
[10] Deng, J., Jiang, Y., Mao, Z.: An artificial immune network approach for pattern recognition. In: Third International Conference on Natural Computation (ICNC 2007) (2007)
[11] Fu, J., Li, Z., Tan, H.-Z.: A hybrid artificial immune network with swarm learning. In: International Conference on Communications, Circuits and Systems (ICCCAS 2007) (2007)
[12] Bereta, M., et al.: Comparing binary and real-valued coding in hybrid immune algorithm for feature selection and classification of ECG signals. Engineering Applications of Artificial Intelligence 20, 571–585 (2007)
[13] Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical advances in artificial immune systems. Theoretical Computer Science 403, 11–32 (2008)
[14] Timmis, J., Andrews, P., Owens, N., Clark, E.: An interdisciplinary perspective on artificial immune systems. Evolutionary Intelligence 1, 5–26 (2008)
[15] Hart, E., Timmis, J.: Application areas of AIS: the past, the present and the future. Applied Soft Computing 8, 191–201 (2008)
[16] Dasgupta, D., et al.: Recent Advances in Artificial Immune Systems: Models and Applications. Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc, 08.24

Novel Binary Encoding Differential Evolution Algorithm

Changshou Deng, Bingyan Zhao, Yanling Yang,
Hu Peng, and Qiming Wei

School of Information Science and Technology, JiuJiang University, 332005 Jiujiang, China
{csdeng,yangyanling}@jju.edu.cn

Abstract. Differential Evolution (DE) algorithm is a successful optimization method in continuous space and has been successfully applied in many different areas. The operators used in DE are simple, however, the mechanism in which the operators are defined, makes it impossible to apply the standard DE directly to the problems in binary space. A novel binary encoding DE (BDE) was proposed to extend DE for solving the optimization problems in binary space. A mixed expression, which constitutes of an arithmetical expression and a logical expression, was used to construct a new mutation operator. And then with a predefined probability, the result of the mutation operator was flipped. Initial experiment results indicate the novel BDE is useful and effective.

Keywords: Discrete optimization, mutation operator, arithmetical expression, logical expression.

1 Introduction

Differential Evolution (DE), proposed by Storn and Price (1997) [1], is a simple yet competitive algorithm for real parameter optimization. DE, just as most popular Evolutionary Algorithms, is a population-based method. DE generates trials by adding the scaled difference of two randomly selected vectors to the third individual. Then DE recombines the trial with its parent using a curtain probability to generate its offspring. In addition, DE employs a one-to-one spawning logic which allows replacement of an individual only if the offspring outperforms its corresponding parent.

Since its inception, DE has been widely used in a variety of continuous fields. For example, Chiou et al. (2004) applied DE to power electronics [2]. Karaboga and Cetinkaya (2006) designed a filter with DE [3]. In Tirronen (2008) [4], a DE-based algorithm is implemented to design a digital filter for paper industry applications. An application to highway network capability optimization is given in Koh (2009) [5]. It has also been successfully used in chemical engineering, machine intelligence and pattern recognition [6].And it has been shown to perform better than the Genetic Algorithm (GA) or the Particle Swarm Optimization (PSO) over several numerical benchmarks [7].

Despite the simplicity and successful application in many engineering fields, its application in the domain of discrete optimization problems is still unusual. This limitation is mainly caused by the working mechanism of DE (which is based on real vectors) and the lack of a principled generalization of DE [8]. In this paper, a novel binary encoding DE was proposed.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 416–423, 2011.
© Springer-Verlag Berlin Heidelberg 2011

The rest of the paper is organized as follows. Section 2 gives a brief introduction of conventional DE. A novel BDE with binary mutation is presented in section 3. The three knapsack Problems are used to evaluate the BDE in section 4. Section 5 concludes this paper.

2 Conventional DE

The DE algorithm is a kind of floating-point encoding evolutionary optimization algorithm. At present, there have been several variants of DE [1]. One of the most promising schemes, DE/Rand/1/Bin , is presented in great detail. It is supposed that we are going to find the minimization of the objective function $f(x)$.

2.1 Generation of Initial Population

The DE algorithm starts with the initial target population $X = (x_{ij})_{m\times n}$ with the size of m and the dimension of n , which is generated by the following way.

$$x_{ij}(0) = x_j^l + rand(0,1)(x_j^u - x_j^l) . \tag{1}$$

where $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$, x_j^u and x_j^l denotes the upper constraints and the lower constraints respectively.

2.2 Mutation Operator

For each target vector $x_i (i = 1,2,\cdots,m)$, a mutant vector is produced by

$$h_i(t+1) = x_{r1} + F(x_{r2} - x_{r3}) . \tag{2}$$

where $i, r_1, r_2, r_3 \in \{1,2,\ldots,m\}$ are randomly chosen and must be different from each other. And F is the scaling factor for the difference between the individual x_{r2} and x_{r3} .

2.3 Crossover Operator

DE employs the crossover operator to add the diversity of the population. The approach is given by (3).

$$u_i(t+1) = \begin{cases} h_i(t+1), & if \quad rand \le CR \ or \ j = rand(i) \\ x_i(t), & otherwise \end{cases} . \tag{3}$$

where $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$, $CR \in [0,1]$ is the crossover probability and $rand(i) \in (1,2,\cdots n)$ is the randomly selected number. The crossover operator can ensure at least one component of the trial individual comes from the mutation vector.

2.4 Selection Operator

To decide whether the trial individual $u_i(t+1)$ should be a member of the next generation, it is compared to the corresponding $x_i(t)$. The selection operator is based on the survival of the fitness among the trial individual and the corresponding one such that:

$$x_i(t+1) = \begin{cases} u_i(t+1), & if \quad f(u_i(t+1)) < f(x_i(t)) \\ x_i(t), & otherwise \end{cases} \tag{4}$$

In short, DE can adapt itself during the search process and find the optimum efficiently and effectively.However, the search set in the continuous domain can satisfy a closure under the mutation operation shown as formula (2). When this mutation operator is employed in the binary space, this closure cann't be satisfied.

3 Binary Encoding DE

It can be easily found that the mutation operator, denoted by the formula (2), can only be effective in real domain. When it is used for the optimization problems in the binary space, the results of the mutation operator may fall out the space. Thus the conventional DE cannot be used in discrete optimization problems directly. A Binary encoding DE with a newly defined mutation operator was proposed to deal with this problem.

3.1 New Mutation Operator

The main nature of the conventional DE is to use the difference of the between vectors to perturb one certain vector. The difference of two multi exclusive vectors is got by the mutation operator in formula (2). For the optimization problems in binary space, the elements of each vector can be regarded as logical number. Borrowing the concept of difference in continuous field, the difference of two binary numbers can be described as the result of exclusive or (XOR) of two binary digits. For instance, the truth table of XOR is shown in table 1.

Table 1. XOR Truth Table

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

In table 1, a simple way to state the exclusive disjunction is "one or the other but not both." When the X and Y are not identical, the output is 1. Hence this can be used to express the difference of the X and Y. This paper will use this to adapt the conventional DE mutation operator.

A new mutation operator is given as formula (5).

$$h_{ij}(t+1) = MOD\ (x_{r1j} + (x_{r2j} \quad XOR \quad x_{r3j}) \tag{5}$$

where MOD is the function of Addition Modulo 2.

Then with a very small probability p, $h_{ij}(t+1)$ was flipped as formula (6).

$$h_{ij}(t+1) = 1 - h_{ij}(t+1) \tag{6}$$

Since the crossover and selection operator in the conventional DE already manipulate the vectors in discrete form, the BDE can inherit them directly.

3.2 Flowchart of the BDE

The main steps of the proposed BDE are as fellows.

Step1: Generation of Binary Initial Population;
Step2: New mutation operator as formula (5) and formula (6);
Step3: Crossover operator as formula (3);
Step4: Selection operator as formula (4)

The flowchart of the BDE is presented in Fig. 1.

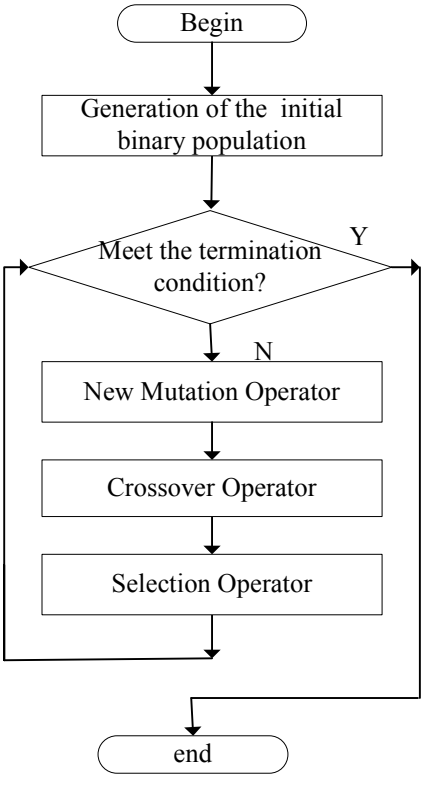


Fig. 1. The flowchart of BDE

4 Numerical Experiments

In this section, the three 0-1 knapsack problems in [9, 10] were used to test the proposed BDE initially.

Since the 0-1 knapsack problems are constrained, the method in [9] was used to deal with the infeasible solutions as well.

The parameters of the BDE are given in table 2.

Table 2. Parameters of the BDE for the three KPs

Examples	Population size	p	CR	Maxgen
Kp1	20	0.005	0.5	50
Kp2	20	0.005	0.5	300
Kp3	20	0.005	0.5	500

For each of the three knapsack problems, 50 trials have been conducted and the best results (Best), average results (Avg), worst results (Worst), standard deviations (Dev) and the success rate (SR) of the BDE are shown in Table 3.

Table 3. Statistical results of the three knapsack problems

Examples	Best	Avg	Worst	Dev	SR
Kp1	1042	1041.8	1037	0.9897	96%
Kp2	3119	3118.3	3113	1.512	68%
Kp3	26559	26553	26535	10.2884	72%

The results in table 3 show us that BDE can find the optimums of all the three Knapsack Problems with comparable small size of population.

The best results compared with TGBDE [8] and several versions of GA [10] are also reported in table 4.

Table 4. Compared Results with other Algorithms

Algorithm	KP1	KP2	KP3
TGA[10]	1042	3077	25848
GGA[10]	1042	3112	26559
HGA[10]	1037	3103	26487
SEGA[10]	1042	3116	26559
TGBDE[8]	1042	3103	26539
BDE	1042	3119	26559

Table 4 shows that for the three Knapsack Problems, the BDE is the only method which can find the best known optimum of each problem. Particularly, for KP2, the best result found by the BDE is the best one among the six different algorithms.

In order to compare the convergence speed of the BDE with that of TGBDE [8], Using the identical initial poppulation, one run was excuted for the above each knapsack problem. The best solution against generation was recorded, as shown in Fig.2.-Fig.4. respectively.

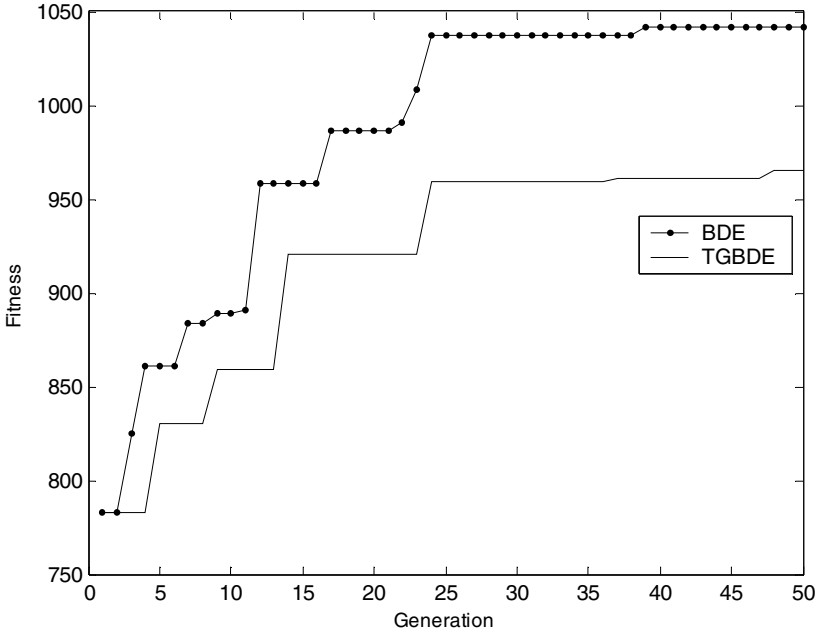


Fig. 2. Best against generation for KP1

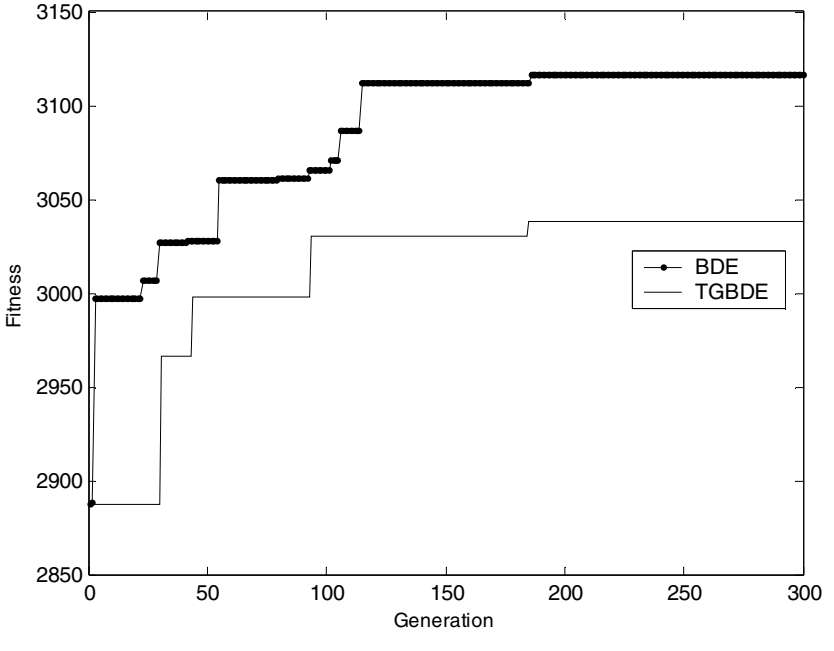


Fig. 3. Best against generation for KP2

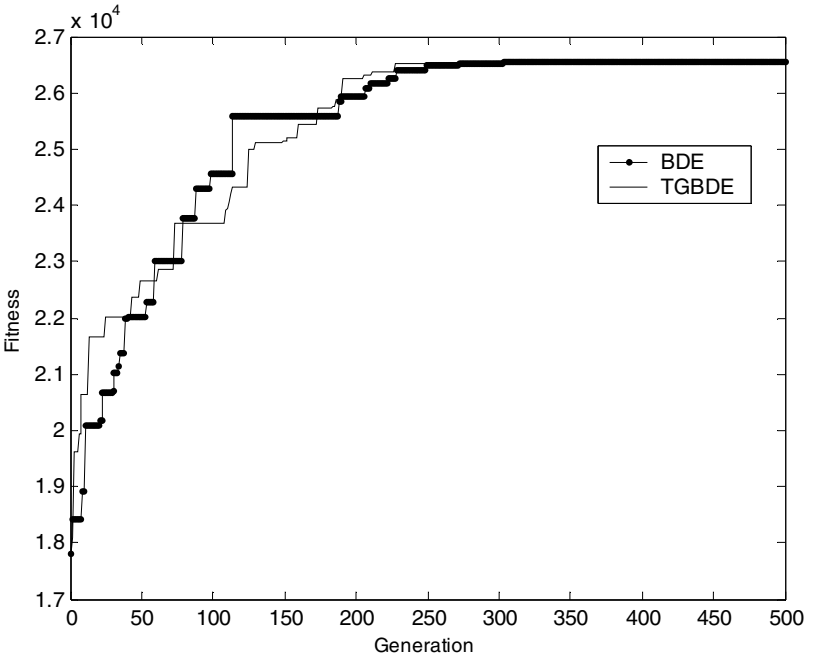


Fig. 4. Best against generation for KP3

A conclusion can be drawn from the Figs 2, 3 and 4, that for the three knapsack problems used in the experiment, the convergence performance of the BDE is slightly better than that of the TGBDE.

Further to evaluate the sensitivity of the probability to flip the digit, five different parameters of p are used to solve Kp2. Then 10 trials have been conducted and the best results (Best), average results (Avg), worst results (Worst), standard deviations (Dev) and the success rate (SR) of the BDE are shown in Table 5.

Table 5. Five different p for Kp2

p	Best	Avg	Worst	Dev	SR
0.1	3118	3116.7	3116	0.9487	0
0.05	3119	3118.4	3116	0.9661	60%
0.025	3119	3118.7	3116	0.9487	90%
0.0125	3119	3117.2	3113	2.8983	70%
0.001	3119	3118.7	3117	0.6749	90%

The results in table 5 show us that the smaller value of the probability p, the better results can be found by the BDE.

5 Conclusions

DE is a recently developed heuristic algorithm that has been empirically proven to be very efficient for global optimization over continuous spaces. In order to extend the field of DE from the continuous domain to the binary domain, a novel BDE was proposed. In the BDE, a new mutation operator was defined to deal with binary digits. Initial experiments on the three different sizes of Knapsack Problems show it is an effective and efficient way to solve the binary optimization problems. In the future, we will investigate this novel BDE to solve other combinatorial problems such as TSP.

Acknowledgments. This work is partially support by the science and Technology Foundation of Jiangxi Province, China under grant no. GJJ10616 &GJJ11616.

References

1. Storn, R., Price, K.: Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 241–354 (1997)
2. Liu, B., Wang, L., Jin, Y.H., et al.: Designing neural networks using hybrid particle swarm optimization. LNCS, pp. 391–397. Springer, Berlin (2005)
3. Rogalsky, T., Derksen, R.W., Kocabiyik: Differential evolution in aerodynamic optimization. In: Proc. of 46th Annual Conf. of Canadian Aeronautics and Space Institute, pp. 29–36 (1999)
4. Das, S., Konar, A.: Design of two dimensional IIR filters with modern search heuristics: a comparative study. International Journal of Computational Intelligence and Applications 6(3), 329–355 (2006)
5. Wang, F.-S., Jang, H.-J.: Parameter estimation of a bio-reaction model by hybrid differential evolution. In: Proc. of the IEEE Congress on Evolutionary Computation, USA, pp. 410-417 (2000)
6. Omran, M., Engelbrecht, Konar, A.: Differential evolution methods for unsupervised image classification. In: Proc. Seventh Congress on Evolutionary Computation. IEEE Press, Los Alamitos (2005)
7. Versterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithm on numerical benchmark problems. In: Evolutionary Computation, CEC 2004, vol. 2, pp. 1980–1987. IEEE press, Portland (2004)
8. Gong, T., Tusion, L.: Differential Evolutionfor Binary Encoding. In: Soft Computing in Industrial Applications, pp. 251–262 (2007)
9. Deng, C., Zhao, B., Yang, Y., Deng, A.: Hybrid Differential Evolution for Knapsack Problem. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 505–512. Springer, Heidelberg (2010)
10. Liu, Y., Liu, C.: A Schema-Guiding Evolutionary Algorithm for 0-1 Knapsack Problem. Iacsit-sc. In: 2009 International Association of Computer Science and Information Technology –Spring Conference, pp. 160–164. IEEE Press, Singapore (2009)

Adaptive Learning Differential Evolution for Numeric Optimization

Yi Liu, Shengwu Xiong, Hui Li, and Shuzhen Wan

School of Computer Science and Technology,
Wuhan University of Technology, Wuhan 430070, China

Abstract. Differential Evolution algorithm is a simple yet reliable and robust evolutionary algorithm for numeric optimization. However, fine-tuning control parameters of DE algorithm is a tedious and time-consuming task thus became a major challenge for its application. This paper introduces a novel self-adaptive method for tuning the amplification parameters F of DE dynamically. This method sampled appropriate F value from a probabilistic model build on periodic learning experience. The performance of proposed MSDE is investigated and compared with other state-of-art self-adaptive approaches. Moreover, the influence of learning frequency of MSDE is investigated.

Keywords: Differential Evolution, self-adaptive, learning experience.

1 Introduction

Differential evolution (DE) algorithm, proposed by Storn and Price[1], is a simple but powerful population-based stochastic search technique for solving global optimization problems. However, the performance-critical control parameters and learning strategies involved in DE are highly dependent on the problems under consideration. For a specific task, we may have to spend huge amount of time to try through various strategies and fine-tune the corresponding parameters, and this trial-and-error process can be fallible due to the nature of human intervention. This dilemma motivates us to develop a Self-adaptive DE algorithm to solve general problems more efficiently. Recently, several Self-adaptive DE algorithms have been proposed to tackle this problem. Omran[3] proposed a Barebones DE which generate F and P_r value from a preset fine-tuned normal distribution. Qin[4] proposed SaDE algorithm which self-adapt suitable learning strategy according to the learning experience. Abbass[5].proposed a SPDE algorithm where P_r will be updated according to arithmetic combination of target vector and differential vector of P_r value. Salman [6] investigated and compared the above-mentioned algorithms empirically.

This paper proposed a novel self-adaptive method for tuning the amplification parameters of DE dynamically. Inspired by Estimation of Distribution Algorithm approach[7],the proposed method sampled appropriate amplification factor value from a probabilistic model build on previous learning experience which accumulated periodically and we name the proposed self-adaptive learning algorithm My Self-adaptive Differential Evolution algorithm(referred to as MSDE in remainder of

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 424–430, 2011.
© Springer-Verlag Berlin Heidelberg 2011

paper).Preliminary test results show that MSDE algorithm has promising performance and is competitive to other well-known self-adaptive approaches.

The reminder of the paper is organized as follows: Section 2 provides a brief description of DE. The proposed MSDE is given in Section 3. Results of the experiments are presented in Section 4. Finally, Section 5 concludes the paper.

2 Basic Differential Evolution

Basic DE algorithm comprise of three major operators, namely mutation, crossover and selection. They are described briefly as followed:

Mutation: According to the mutation operator of the most commonly used DE/rand/1/bin strategy [1], for each target individual $x_i(t)$, a mutant vector $v_i(t)$ is determined by the following equation,

$$v_i(t) = x_{i1}(t) + F(x_{i2}(t) - x_{i3}(t)) \quad . \tag{1}$$

Where i_1, i_2, i_3 are index randomly selected from the set $\{1,2,\ldots,NP\}$: Note that index must be mutually different from each other and from the running index i so that population size NP must be at least four.

Crossover: DE follows a discrete recombination approach where elements from the parent vector $x_i(t)$ are combined with elements from the trial vector $v_i(t)$ to produce the offspring $\mu_i(t)$ .Using the binomial crossover,

$$\mu_{ij}(t) = \begin{cases} v_{ij}(t) \text{ if } U(0,1) < P_r \text{ or } j = r, \\ x_{ij}(t) \text{ otherwise,} \end{cases} \tag{2}$$

Even when $P_r = 0$, at least one of the parameters of the offspring will differ from the parent(forced by the condition $j=r$).

Selection: To decide whether the trial vector $v_i(t)$ should be a member of the population comprising the next generation, it is compared to the corresponding target vector $x_i(t)$; and the greedy selection strategy is adopted in DE. The selection operator is as following,

$$x_i(t+1) = \begin{cases} \mu_i(t), f(\mu_i(t)) < f(x_i(t)) \\ x_i(t), \text{otherwise} \end{cases} \tag{3}$$

The above 3 steps are repeated generation after generation until a pre-specified stopping criteria is satisfied.

3 My Self-adaptive Differential Evolution

3.1 Learning Strategy

In this paper, we incorporated a learning frequency parameter L into proposed MSDE (Later experiment section show MSDE is insensitive to this parameter). L will determine the periodic learning span during the whole search. For every learning span

(L generations), the F value that successfully entering next generation will be collected in a F value pool. We pool all the successful F value together as a repository of learning sample and build a probabilistic model of its value distribution periodically. New F values will be generated according to this probabilistic model during the next L generation hence the value will adapt itself in accordance with different stages of search process. We assume F is normal distributed and the mean μ_m and σ_m of this distribution is dependent on aggregated F value within pool. This normal distribution parameters will remain the same for several generations and then a new set of μ_m and σ_m is recalibrated on update generation. An update generation is the generation where μ_m and σ_m recalculation will be performed. In some rare case where F value pool is empty, F value will be generated according to initial distribution *N(0.7, 0.3)*.After each update generation, the accumulated F value pool will be clear in case of previous outdated learning influence, consequently, $f_{i,t}$ will be generated for each individual i according to,

$$f_{i,t} = N(\mu_m, \sigma_m) \quad . \tag{4}$$

t denote tth generation, μ_m, σ_m are mean and standard deviation of aggregated F value in previous L learning generations and the initial F value of every individual is generated from normal distribution *N(0.7, 0.3)*. Parameter μ_m, σ_m will be update at each update generation.

The crossover probability P_r of MSDE will be generated according to

$$P_{r,i} = N_i(0.5,0.15) \quad . \tag{5}$$

the subscript i denote by P_r value is sampled anew for each individual and this will generate P_r values which fits well within the range [0,1) [6].

4 Experimental Comparison

4.1 Experimental Setting

In this section, two well-known self-adaptive DE algorithm Barebones DE and Self-adaptive Pareto DE are chosen as competing candidate. No attempt was made to tune the MSDE parameters to each problem. The rationale behind this decision is the fact that in real-world applications the evaluation time is significant and as such parameter tuning is usually a time consuming process. 4 Multimodal functions and 1 unimodal functions are selected for performance assessment. Test function Rosenbrock,Ackey,Rastrigin, Griewank's function definition and variable X bounds were taken from [8],function F9's definition and X bounds came from [9]. Asymmetric Initialization method proposed by Angeline [10] is adopted in experiment to avoid accidently generating a near-optimum initial solution. The average results of 20 independent runs with different random seed are summarized in Table 1.

For all the algorithms used in this section, population size *NP=20*. All functions were implemented in 20 dimensions except for the two-dimensional F9 function. All

results are obtained through *T=100000* function evaluations except for F9 function where *T=100000*.

PC Configuration: AMD Phenom II X4 3Ghz processor with 2Gb memory and Visual C++ with boost library.

Parameter Configuration: The learning frequency parameter *L* of msDE is fixed to 200 in all the function except for Function F9 where *L=50*.

P_r, F in bbDE is generated according to [2].

spDE's parameter is set(all the operators dealing with multi-objective functions were removed) as to [5] accordingly.

Table 1. Mean and standard deviation(±SD) of test function optimization result, upper row is mean value, lower row is standard deviation

Fun. / Algo.	F9	Rastrigin	Rosen-brock	AckeyF1	Griewank
bbDE	-0.9973	0.1492	4.0188	5.596e-015	**0**
	0.0036	0.3553	1.4026	7.952e-015	**0**
msDE	**-0.9979**	**6.631e-013**	7.9386	**3.109e-015**	0.0023
	0.0034	**2.888e-012**	1.4701	**0**	0.0045
spDE	-0.9961	0.049748	**1.0734**	9.681e-015	**0**
	0.0043	0.216846	**1.0147**	7.476e-015	**0**

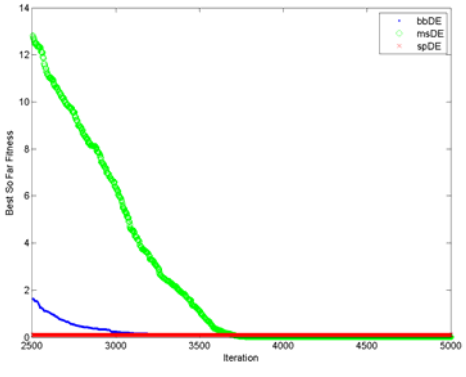


Fig. 1. Function Rastrigin: fitness value comparison, averaged over 20 runs

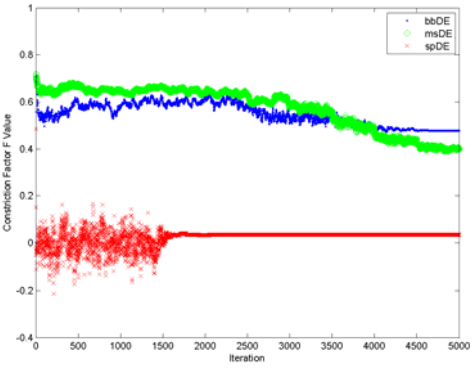


Fig. 2. Function Rastrigin: *F* value comparison, averaged over 20 runs

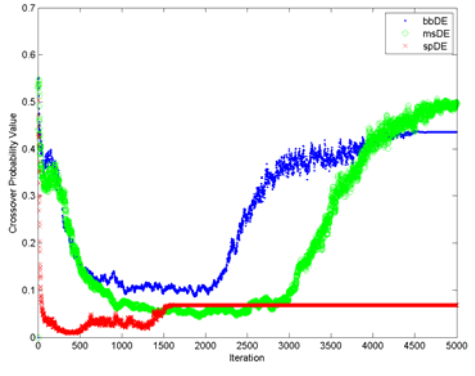


Fig. 3. Function Rastrigin: Pr value comparison, averaged over 20 runs

Table 1 show that msDE outperform bbDE and spDE on 3 functions out of 5 total test functions. MSDE outperform bbDE and spDE on function Rastrigin significantly and it also has higher mutation rate than that of bbDE and spDE on the same function(Fig.3) while bbDE and spDE outperform msDE in unimodal function Rosenbrock and multimodal function Griewank.Since Griewank function with high dimensionality(≥ 10) is unimodal function essentially,we may conclude that msDE's convergent rate is slower than that of bbDE and spDE but it has better performance on multimodal functions than bbDE and spDE .

4.2 Impact of Learning Frequency *L*

In this section, F9 function and 5 different *L* values are chosen to illustrate the influence of learning frequency *L* on algorithm performance and the experimental results are listed below:

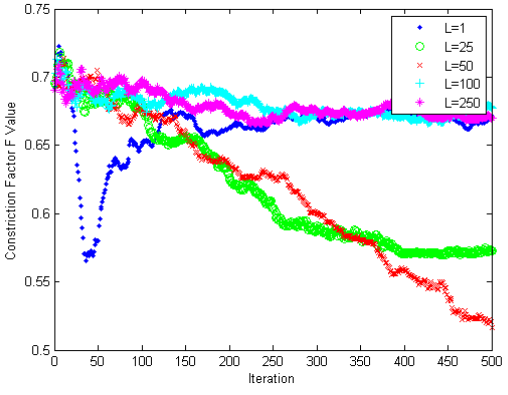


Fig. 4. Generational comparison of different L value on Function F9

Table 2. Mean and standard deviation (SD) of function F9 optimization result

Value / *L*	Fitness mean std	*F* mean std
1	-0.9972 0.0036	0.6620 0.0235
25	-0.9982 0.0030	0.6195 0.0450
50	-0.9985 0.0022	0.6162 0.0549
100	-0.9968 0.0037	0.6792 0.0084
250	-0.9976 0.0028	0.6791 0.0093

Additional 1 degree freedom two-tailed t-test with confidence level 0.95 show that no statistically significant differences between all 5 *L* values as far as mean of fitness (list table above) is concerned. That means MSDE is insensitive to parameter *L* on function F9.Similar result can be observed on other 4 test functions(not listed due to space limit).Thus we can conclude that MSDE is relatively insensitive to its learning frequency parameter *L*.

5 Conclusion

In this paper we proposed a self-learning MSDE algorithm which sampled appropriate *F* value from a probabilistic model build on previous learning experiences. The performance of proposed algorithm is investigated and compared with other well-known self-adaptive algorithms. Preliminary experimental results show MSDE is competitive to bbDE and spDE. Moreover, the influence of learning frequency *L* of MSDE is investigated.

For future work, we intend to further examine convergence rate of MSDE algorithm and compared its various performance index to other EA algorithms.

Acknowledgements

This work was supported in part by the National Science Foundation of China under grant no. 40971233.

References

1. Storn, R.: On the usage of differential evolution for function optimization. In: The North American Fuzzy Information Processing Society Conference, Berkeley, pp. 519–523 (1996)
2. Omran, M.G.H., Engelbrecht, A.P., Salman, A.: Differential Evolution Based Particle Swarm Optimization Swarm Intelligence Symposium, SIS 2007, pp. 112–119 (2007)
3. Omran, M.G.H., Salman, A., Engelbrecht, A.P.: Self-Adaptive Differential Evolution. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 192–199. Springer, Heidelberg (2005)
4. Qin, A.: Self-adaptive Differential Evolution Algorithm for Numerical Optimization. In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791 (2005)
5. Abbass, H.: The self-adaptive pareto differential evolution algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 831–836 (2002)
6. Omran, M.G.H., Engelbrecht, A.P., Salman, A.: Empirical Analysis of Self-Adaptive Differential Evolution. Accepted for European Journal of Operational Research (2006)
7. Sun, J., Zhang, Q., Tsang, E.: DE/EDA: New Evolutionary Algorithm for Global Optimisation. Information Sciences (169), 249–262 (2005)
8. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2), 82–102 (1999)
9. Whitley, Mathias, K., Rana, S.: Building better test functions. In: Proceedings of the Sixth Int. Conf. on Genetic Algorithms, pp. 239–246 (1995)
10. Angeline, P.: Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)

Differential Evolution with Improved Mutation Strategy

Shuzhen Wan, Shengwu Xiong*, Jialiang Kou, and Yi Liu

School of Computer Science and Technology,
Wuhan University of Technology, Wuhan 430070, China
wanshuzhen@163.com

Abstract. Differential evolution is a powerful evolution algorithm for optimization of real valued and multimodal functions. To accelerate its convergence rate and enhance its performance, this paper introduces a top-p-best trigonometric mutation strategy and a self-adaptation method for controlling the crossover rate (CR). The performance of the proposed algorithm is investigated on a comprehensive set of 13 benchmark functions. Numerical results and statistical analysis show that the proposed algorithm boosts the convergence rate yet maintaining the robustness of the DE algorithm.

Keywords: trigonometric differential evolution, differential evolution, benchmark function, crossover operation.

1 Introduction

Differential evolution[1] has become a popular algorithm for global optimization in various fields. It is a simple yet powerful population-based stochastic search technique which has shown superior performance not only in benchmark functions but also in real-world application[2]. It has been shown to perform better than genetic algorithm (GA)[3], and particle swarm algorithm (PSO)[4] over several benchmarks[5]. In DE, there exist many mutation strategies out of which a few can solve some particular problems. Moreover, three control parameters involved in DE, i.e., population size NP, scale factor F , crossover rate CR , will significantly influence the optimization performance of DE. A lot of research has been focused on these issues which can be found in [6]. Experimental parameter study and empirical parameters setting of DE have been carried out in [7]. The analysis of relationship between population diversity and control parameters have been investigated in [8].

In [9], Fan and Lampinen proposed a DE variant, namely trigonometric mutation differential evolution(TDE), in which, a new local search operation, trigonometric mutation, was proposed and embedded into DE. The trigonometric mutation is a rather greedy operator which can greatly speed up the convergence rate of DE. However, such an operator usually leads to a greedy algorithm prone to converging prematurely into local optima. In the TDE algorithm, the parameter M_t (trigonometric mutation

* Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 431–438, 2011.
© Springer-Verlag Berlin Heidelberg 2011

probability) can maintain good balance between convergence rate and quality of solutions. Therefore, TDE has better convergence rate and can obtain an acceptable solution over some benchmark functions as well. Fan and Lampinen also point out that the trigonometric mutation operation biases the new trial individual strongly in the direction where the best one among three individuals chosen for the mutation is. In TDE, Whether the new trial individual will be prone to the best individual or to the poor individual depends on choosing of the donor individuals. If there one donor individual comes from the top $p\%$ of best individuals, the new trial individual will be prone to those individuals, and the population will converge to top $p\%$ of the best individuals. Thus, the convergence rate will be higher (proved in our experiments). But the modified trigonometric mutation operator has the danger of making the population converge to local optima. To avoid this phenomenon happening, a good number of techniques have been proposed to tune the mutation strategy and parameters such as population size NP [10] , the scale factor F , and the crossover rate CR [11, 12]. Brest et al. presented a variant of DE called JDE, using self-adaptive updating strategy of the parameter F and CR [13]. The simulation results show the JDE has better performance than some evolutionary algorithms, like FEP and CEP over 21 benchmark functions.

In our paper, we introduce a top-p-best trigonometric mutation to DE to speed up the convergence process. In this method, one of the donor vectors comes from the top $p\%$ best individuals. In order to improve the population diversity and avoid trapping in local optima, we adopt the random self-adaptive parameter setting strategy of CR [13]. Computational experiments and comparisons show that the proposed algorithm performs better than TDE, standard DE and JDE[13], when applied to 13 well-known numerical benchmark functions.

The rest of this paper is organized as follows: the conventional DE, TDE are reviewed in sections 2 and 3, respectively; section 4 describes the proposed algorithm; in section 5, experimental studies are present; in section 6, finally conclusions are drawn.

2 Differential Evolution

There are several variants of DE[1]. In this paper, we shall follow the version of *DE/rand/1/bin*. This particular scheme is described as follows:

Like any other evolutionary algorithm, DE starts with a population of NP candidate solutions which can be represented as the D-dimensional parameter,

$X_{i,G}, i = 1, 2, 3, ..., NP$, where i index denotes the i th individual of the population, G denotes the generation to which the population belongs. NP is the number of members in a population. Successive populations are generated by adding the weighted difference of two randomly selected vectors to a third randomly selected vector. This operation is the mutation operator, and then crossover operator is employed to generate new candidate vectors. A selection scheme is employed to determine whether the offspring or the parent survives to the next generation. The process is repeated until a termination criterion is reached. The details are described as follows.

2.1 Mutation

The mutation operation of DE applies the vector differentials between the randomly selected vectors from the population for making a perturbation to the individual subject to the mutation operation. At generation G, for the i th target individual $X_{i,G}$, the perturbed individual $V_{i,G}$ is generated based on the three randomly selected individuals as follows:

$$V_{i,G+1} = X_{r3,G} + F * (X_{r1,G} - X_{r2,G}) \tag{1}$$

Where $i = 1,...,NP, r_1, r_2, r_3 \in \{1,...,NP\}$ are randomly selected and satisfy: $r_1 \neq r_2 \neq r_3 \neq i, F \in [0,1]$, where F is the scale factor.

2.2 Crossover

The perturbed individual, $V_{i,G+1} = (v_{1,i,G+1},...,v_{n,i,G+1})$ and the target population member, $X_{i,G} = (x_{1,i,G},...,x_{n,i,G})$, are then subject to crossover operation to generate the population of candidate or "trial" vectors, $U_{i,G+1} = (u_{1,i,G+1},...,u_{n,i,G+1})$, as follow:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if rand}_j \leq CR \text{ or } j = k \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{2}$$

Where $j = 1,...,n, k \in \{1,...,n\}$ is a random parameter's index, chosen for each i, and the crossover rate, $CR \in [0,1]$, the other control parameter is set by the user.

2.3 Selection

The selection operation selects, according to fitness value of the population vector and its corresponding trail vector, which vector will become a member of the next generation. If we have the minimization problem, the following selection rule is used:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & \text{if } f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \tag{3}$$

3 TDE

In TDE algorithm[9], the trigonometric mutation operation is embedded into the basic DE to improve the convergence rate. The main difference between DE and TDE is the

way the mutation operation is performed. The trigonometric mutation operation is performed according to the following formulation:

$$\begin{aligned} V_{i,G+1} = &\left(X_{r1,G} + X_{r2,G} + X_{r3,G}\right)/3 + \left(p_2 - p_1\right)\left(X_{r1,G} - X_{r2,G}\right) \\ &+ \left(p_3 - p_2\right)\left(X_{r2,G} - X_{r3,G}\right) + \left(p_1 - p_3\right)\left(X_{r3,G} - X_{r1,G}\right) \end{aligned} \tag{4}$$

Where for $i = 1, 2, 3$,

$$p_i = \frac{\left|f\left(X_{i,G}\right)\right|}{\left|f\left(X_{1,G}\right)\right| + \left|f\left(X_{2,G}\right)\right| + \left|f\left(X_{3,G}\right)\right|}$$

Thus, the trigonometric mutation operation is a greedy operator that for three given points generates an offspring by exploiting the most promising directions. The performance of this operator can offer an exploitative alternative to DE. The trigonometric mutation biases the offspring strongly in the optimal directions, so the convergence rate will be accelerated. However, such an operator usually leads to the greedy algorithm prone to converge prematurely into a local optimum, there need some methods to avoid this happen. In TDE, the mutation probability M_t is used for this purpose.

4 Modified Trigonometric DE

It has shown that TDE has the rapid convergent process because the trigonometric mutation is a rather greedy operator, which biases the new trial solution in the direction of the best one among three donor individuals. Inspired from this, we assume that if one of donor individuals is come from the top $p\,\%$ best individuals within the population, the others are from 1- $p\,\%$ members of the population, the new trial vector will converge to top $p\,\%$ individuals, so the convergence rate will be higher. We call this DE variant as top-p-best TDE. The following experiments are partly proven our idea. But the experiments show that though top-p-best TDE accelerates the convergence rate, the population may prematurely lose diversity and converge to the local optima. The previous statements have been shown that JDE has very strong robustness because of its self-adaptive control parameter strategy. Compare with other self-adaptive scheme [14], this technique is simple yet effective. We apply this strategy into the crossover rate setting of our algorithm to increase the diversity of the population and avoid prematurity. For simplicity, we call our algorithm as WDE. The outline of our algorithm as follows:

(1) Initialize the population, choose F and M_t , and determine the initial CR .

(2) Mutation operation:

(2.1) Perform p-best-trigonometric mutation with Equation (4) with a probability M_t (in which, one of the donor individual comes from the top $p\,\%$ best individuals of the population).

(2.2) Perform original DE's mutation with Equation (1) with a probability (1- M_t).

(3) Crossover operation:

(3.1) update the crossover rate with Equation (6).

(3.2) perform the crossover operation with the updated CR .

(4) Evaluate the population with the object function.

(5) Selection.

(6) Repeat step (2) to (5) until the termination criterion is satisfied.

5 Experimental Studies

5.1 Experimental Setup

Experiments were conducted on a set of 13 benchmark functions which are detailed in literatures [13] to evaluate the proposed algorithm and three other DE algorithms. For functions $f_1 - f_{13}$, 30-dimensional (30-D) were tested. The maximum number of function evaluations (FEs) is set to 150 000 for the functions $f_1, f_6, f_{10}, f_{12}, f_{13}$, 200 000 for function f_2, f_{11} , 300 000 for f_7 , 500 000 for f_3, f_4, f_9 , 900 000 for f_8 and 2000 000 for f_5 . All experiments were run 30 times independently. The population size NP is set to 100 for all algorithms. The algorithms in the comparison are listed:

Standard DE (DE/rand/1/bin): $F = 0.5, CR = 0.9$;TDE: $F = 0.5$, $CR = 0.9$, $M_t = 0.05$; [9]Top-p-best TDE: $F = 0.5, CR = 0.9, p = 1, M_t = 0.05$; Our algorithm: $F = 0.5, CR = 0.9, p = 1, M_t = 0.05$; JDE[13]: $F = 0.5$, $CR = 0.9$, $\tau_1 = 0.1, \tau_2 = 0.1$.

5.2 Comparison between WDE and Other DE Algorithms

Table 1 reports the results of 13 benchmark functions over 30 runs. However, it can be easily seen that on average the WDE algorithm outperforms all other compared algorithms.

Table 1. Mean and standard deviation of 13 benchmark functions optimization results averaged over 30 runs, of WDE, top-p-best TDE, TDE, JDE and DE

function	WDE		Top-p-best TDE		TDE		JDE		DE	
	Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev
1	6.60E-48	1.00E-47	2.40E-52	5.20E-52	2.70E-34	5.90E-34	2.70E-28	2.90E-28	1.20E-13	1.50E-13
2	3.60E-40	3.30E-40	2.10E-32	1.10E-31	1.70E-28	1.40E-28	1.90E-23	1.30E-23	9.60E-10	5.60E-10
3	5.70E-20	8.70E-20	3.30E-47	8.90E-47	9.90E-18	3.30E-17	3.90E-05	3.90E-05	5.40E-11	5.20E-11
4	1.20E-17	2.80E-17	8.50E-04	2.20E-03	5.10E+00	1.70E+00	7.80E-10	4.00E-09	3.60E-01	9.40E-01
5	0.00E+00	0.00E+00	8.00E-01	1.60E+00	6.60E-01	1.50E+00	2.50E-29	3.20E-29	3.30E-31	1.80E-30
6	0.00E+00	0.00E+00	6.20E+00	7.10E+00	5.30E-01	1.20E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
7	1.80E-03	4.20E-04	4.10E-03	1.70E-03	2.10E-03	7.30E-04	3.00E-03	8.00E-04	4.70E-03	1.20E-03
8	-1.30E+04	4.10E+01	-1.10E+04	4.30E+02	-1.20E+04	2.20E+02	-1.30E+04	1.90E-12	-1.20E+04	1.40E+02
9	0.00E+00	0.00E+00	3.20E+01	7.50E+00	7.20E+00	3.10E+00	0.00E+00	0.00E+00	8.40E+01	3.60E+01
10	7.40E-15	1.30E-15	1.90E+00	6.00E-01	1.90E-01	3.80E-01	8.30E-15	1.10E-15	1.10E-07	2.40E-08
11	0.00E+00	0.00E+00	2.00E-02	2.00E-02	3.70E-03	6.40E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	1.60E-32	5.60E-48	4.10E-01	5.50E-01	3.50E-03	1.90E-02	8.60E-30	6.00E-30	1.30E-14	1.00E-14
13	1.30E-32	1.30E-32	8.10E-02	3.90E-01	3.70E-04	2.00E-03	1.90E-28	3.10E-28	9.10E-14	7.00E-14

For the unimodal functions f_1, f_3, WDE is surpassed by top-p-best TDE, but performs much better than all other compared algorithms. This shows that top-p-best TDE has the faster convergence rate than WDE on some unimodal functions. The smallest values of functions f_2, f_4 and f_5 are produced by WDE. On functions $f_7, f_{10}, f_{12}, f_{13}$, the WDE algorithm is superior to all other algorithms. On functions f_6, f_8, f_9, f_{11}, there is almost no significant difference between WDE and JDE, but WDE outperforms other compared algorithms. Best results yielded by WDE on functions $f_7, f_{10}, f_{12}, f_{13}$. From Fig.2, we can observe that the proposed WDE algorithm also performs well on the convergence speed. Though the top-p-best TDE has rapid convergence rate on some unimodal test functions, it performs poorly on multimodal functions.

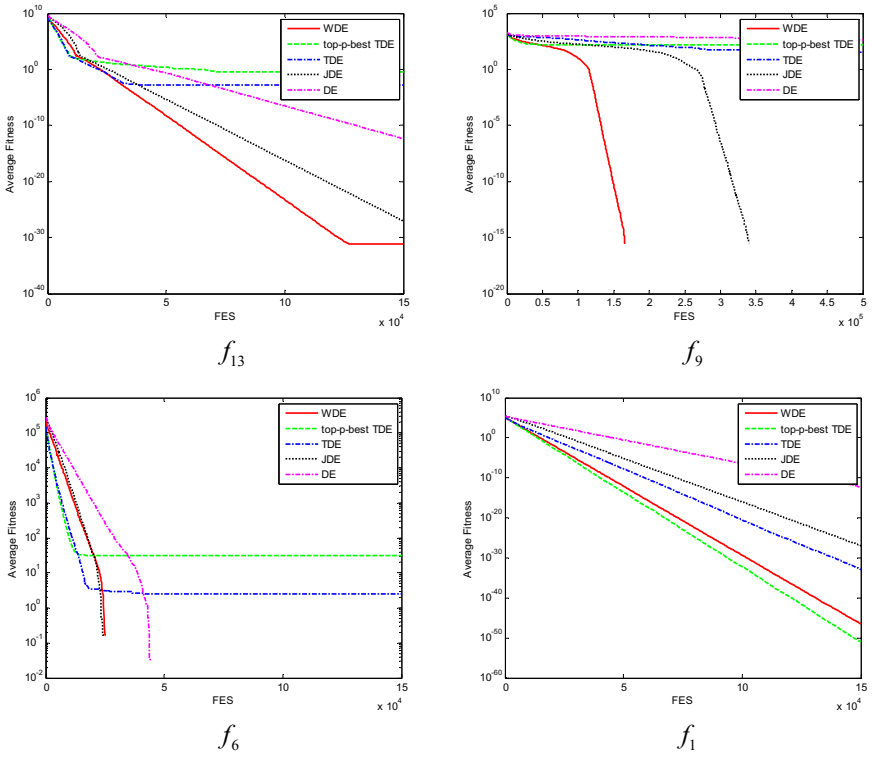


Fig. 1. Average Fitness comparison on functions f_{13}, f_9, f_6, f_1, over 30 runs

5.3 Discussion on Parameter p

The parameter p represents that one of donor individuals comes from top $p\%$ best individuals of population. In order to study this parameter, we set p to 1,5,10 for several benchmark functions. The Fig.2 shows that for test functions f_7, f_2, the mean best

values are smallest for $p=1$, and when $p=10$, the result is worst. However, for functions f_{10}, the values yielded by the proposed algorithm are no significant difference when p is set to 1,5,10. For function f_4, when p is set to 1, 5, the fitness is almost the same, but it becomes worse when p is set to 10. It can be observed that WDE is sensitive to p for some unimodal functions, but insensitive to some multimodal functions. From our experiments, we suggest that the proposed algorithm WDE will produce the best fitness when p is set to 1. It means that all individuals will converge to the best one when using the trigonometric mutation operation.

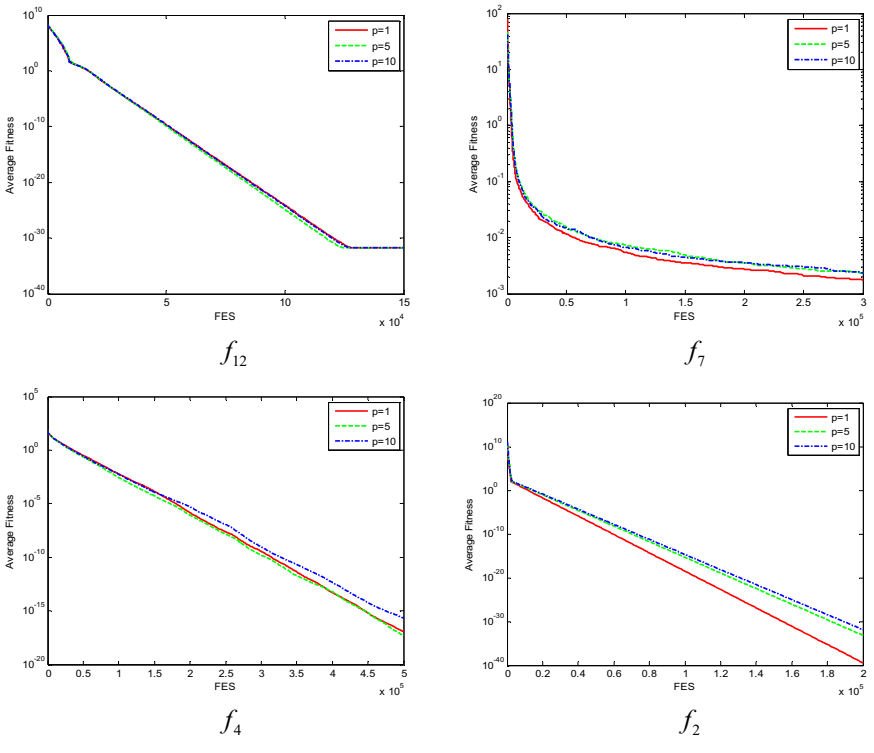


Fig. 2. Average Fitness when p is set to 1,5,10 respectively on functions f_{12}, f_7, f_4, f_1, over 30 runs

6 Conclusion

In this paper, we proposed a new DE variant (WDE) which applied the top-p-best trigonometric mutation strategy and the self-adaptation control parameter strategy for the crossover rate (CR). The new mutation operation helps in accelerating the convergence rate of the proposed algorithm. And the crossover rate update scheme makes a trade-off between the fast convergence rate and trapping in local optima. The proposed

algorithm (WDE) shows better convergence performance than other DE variants in our test over 13 benchmark functions, which are of low or high dimension, unimodal or multimodal, continuous or discontinuous.

References

1. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. J. Global Optim. 11, 341–359 (1997)
2. Babu, B.V., Angira, R.: Modified differential evolution (MDE) for optimization of non-linear chemical processes. Comput. Chem. Eng. 30, 989–1002 (2006)
3. Holland, J.H.: Adaptation in natural and artificial system: an introduction with application to biology, control and artificial intelligence. University of Michigan Press, Ann Arbor (1975)
4. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE, Perth (1995)
5. Vesterstrm, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1980–1987. Electrical and Electronics Engineers Inc. (2004)
6. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. 33, 61–106 (2010)
7. Gamperle, R., Muller, S.D., Koumoutsakos, A.: A parameter study for differential evolution. Advances in Intelligent systems, Fuzzy systems, Evolutionary Computation 10,293–298 (2002)
8. Zaharie, C.: Critical values for the control parameters of differential evolution algorithms. In: 8th International Conference on Soft Computing, pp. 62–67 (2002)
9. Fan, H.Y., Lampinen, J.: A trigonometric mutation operation to differential evolution. J. Global Optim. 27, 105–129 (2003)
10. Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. Soft Computing - A Fusion of Foundations, Methodologies and Applications 10, 673–686 (2006)
11. Gong, W., Fialho, L., Cai, Z.: Adaptive strategy selection in differential evolution. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 409–416. ACM, Portland (2010)
12. Pan, Q.K., Suganthan, P.N., Wang, L., Gao, L.A., Mallipeddi, R.: A differential evolution algorithm with self-adapting strategy and control parameters. Comput Oper. Res. 38, 394–408 (2011)
13. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE T. Evolut. Comput. 10, 646–657 (2006)
14. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1785–1791. IEEE, New York (2005)

Gaussian Particle Swarm Optimization with Differential Evolution Mutation

Chunqiu Wan[1], Jun Wang[2,⋆], Geng Yang[1], and Xing Zhang[3]

[1] Department of Automation, Tsinghua University, Beijing, China
[2] Dept. of Control Science & Engn., Tongji University, Shanghai, China
[3] School of Aerospace, Tsinghua University, Beijing, China
junwang@tongji.edu.cn

Abstract. During the past decade, the particle swarm optimization (PSO) with various versions showed competitiveness on the constrained optimization problems. In this paper, an improved Gaussian particle swarm optimization algorithm (GPSO) is proposed to improve the diversity and local search ability of the population. A mutation operator based on differential evolution (DE) is designed and employed to update the personal best position of the particle and the global best position of the population. The purpose is to improve the local search ability of GPSO and the probability to find the global optima. The regeneration strategy is employed to update the stagnated particle so as further to improve the diversity of GPSO. A simple feasibility-based method is employed to compare the performances of different particles. Simulation results of three constrained engineering optimization problems demonstrate the effectiveness of the proposed algorithm.

Keywords: Gaussian particle swarm optimization, differential evolution, regeneration strategy, feasibility-based comparison method.

1 Introduction

Constrained optimization problems frequently appear in engineering optimization, economic, military, network and management science, etc. It is usually difficult or even impossible for the traditional methods to solve them because they could not provide the derivative information of the objective function and constraints. The evolutionary algorithms (EA) based on population have attracted much attention because they could find high quality solutions without the derivative information [1]. As a new branch of EA, the particle swarm optimization algorithm (PSO) has attracted much attention for its simpleness, efficiency and robustness [2] [3].

There have been many studies for PSO on constrained optimization problems. Takahama [4] [5] introduced the ε-constrained method into PSO so as to find much better solutions for constrained optimization problems and to improve

⋆ Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 439–446, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the efficiency and stability of this algorithm. He [6] and Sun [7] suggested the hybrid PSO with a feasibility-based rule in order to guide the swarm to the feasible region quickly. Huang [8] and He [9] introduced the co-evolutionary mechanism into differential evolution (DE) and PSO so that the solutions and penalty coefficients could evolve interactively and self-adaptively. Liu [10] incorporated DE into PSO to force the population jump out of stagnation.

Recently, Gaussian particle swarm optimization algorithm (GPSO) has shown promising results for solving constrained optimization problems [11]. However, GPSO is weak in the local searching while having high global search ability. The low local search ability may mislead the swarm from the promising region and weaken this algorithm's optimization precision. In this paper, a mutation operator based on differential evolution (DE) is designed for the personal best solutions and the global best solution so as to improve the local search ability of GPSO and find much better solutions for constrained problems. The stagnated particle is regenerated in order to avoid getting trapped into a local optimum. The feasibility-based method is employed to compare the performances of different particles so as to find the feasible solution quickly and improve the optimization efficiency. The improved GPSO is employed to solve three engineering optimization problems to demonstrate its effectiveness.

The remainder of this paper is organized as follows. Section 2 describes the improved GPSO approach. Section 3 presents the optimal results for three engineering problems to demonstrate the effectiveness of the proposed method. Section 4 makes concluding remarks and maps out the directions for future work.

2 The Improved Optimization Algorithm

2.1 Gaussian Particle Swarm Optimization Algorithm

PSO is a population-based, global and stochastic optimization algorithm [12]. It comprises of two parts: position matrix $\mathbf{Z} = [\mathbf{z}_1^{\mathrm{T}}, \cdots, \mathbf{z}_i^{\mathrm{T}}, \cdots, \mathbf{z}_{N_p}^{\mathrm{T}}]^{\mathrm{T}}$ and velocity matrix $\mathbf{V} = [\mathbf{v}_1^{\mathrm{T}}, \cdots, \mathbf{v}_i^{\mathrm{T}}, \cdots, \mathbf{v}_{N_p}^{\mathrm{T}}]^{\mathrm{T}}$. $\mathbf{z}_i (i \in \{1, \cdots, N_p\})$ represents a potential solution of the optimization problem. $\mathbf{v}_i$ represents the modification to solution $\mathbf{z}_i$. "N_p" represents the population size. The position matrix $\mathbf{Z}$ and velocity matrix $\mathbf{V}$ are randomly initialized in the search space. The search for optimal position is carried out by moving toward both their own historical best positions $\mathbf{Z}^p$ and the population's historical best position $\mathbf{z}^g$.

GPSO was proposed by Krohling with the purpose of improving the convergence ability of the algorithm and avoiding the necessity of tuning many parameters like the basic PSO [13] [14]. The only parameter to be specified by the user was the number of particles. Each element of the velocity matrix and position matrix is usually updated by [13] [14]:

$$v_{i,j}(t+1) = v_{i,j}(t) + |g_1| \left(z_{i,j}^p(t) - z_{i,j}(t)\right) + |g_2| \left(z_j^g(t) - z_{i,j}(t)\right) \quad (1)$$

$$z_{i,j}(t+1) = z_{i,j}(t) + v_{i,j}(t+1) \quad (2)$$

where $i \in \{1, \cdots, N_p\}$ is the particle index, $j \in \{1, \cdots, n\}$ is the dimension index, $t \in \{1, \cdots, T_{\max}\}$ is the iteration index, $T_{\max}$ is the maximum number of iterations, $|g_1|$ and $|g_2|$ are the absolute values of random numbers generated according to the normalized Gaussian distribution $N(0, 1)$.

In this paper, if the variable value $v_{i,j}(t+1)$ and $z_{i,j}(t+1)$ generated by Eq. (1) and Eq. (2) violates the boundary constraints, they will be randomly regenerated in the search space of the corresponding variable so that the particles can avoid being concentrated on the boundary and find the global optimum more easily.

2.2 Differential Evolution Mutation

DE was introduced by Storn and Price to solve the optimization problems in continuous spaces [15] [16]. This algorithm uses one main parent and two supportive parents to generate a new offspring. By linear combination of three individuals with certain probability, DE has strong local search ability.

Though GPSO has strong global search ability, its local search ability is weak. In this paper, a mutation operator based on DE is designed and incorporated into GPSO to overcome its weakness so as to improve both the precision and robustness. DE is incorporated into GPSO in the following manner: In the process of optimization, if there is no improvement to the personal best $\mathbf{z}_i^p$ or the global best $\mathbf{z}^g$, mutation is applied to $\mathbf{z}_i^p$ or $\mathbf{z}^g$. As a result, a corresponding trail vector $\mathbf{z}^{new}$ is yielded. Then, $\mathbf{z}_i^p$ or $\mathbf{z}^g$ is compared with $\mathbf{z}^{new}$, and $\mathbf{z}_i^p$ or $\mathbf{z}^g$ would be replaced by $\mathbf{z}^{new}$ if $\mathbf{z}^{new}$ wins.

Assuming that $\mathbf{z}_i^p$ or $\mathbf{z}^g$ is represented by $\mathbf{z}^{old}$, the rule of generation of the trial vector $\mathbf{z}^{new}$ for DE mutation is as follows:

$$\mathbf{z}^{new} = \mathbf{z}_{r1}^p + F(\mathbf{z}_{r2}^p - \mathbf{z}_{r3}^p) \tag{3}$$

$$z_j^{new} = \begin{cases} z_j^{new}, \text{ if } r_{cj} \le P_m \text{ or } j = r_c \\ z_j^{old}, \text{ otherwise} \end{cases} \tag{4}$$

$$z_j^{new} = \begin{cases} z_{\min,j}, \text{ if } z_j^{new} < z_{\min,j} \\ z_{\max,j}, \text{ if } z_j^{new} > z_{\max,j} \\ z_j^{new}, \text{ otherwise} \end{cases} \tag{5}$$

where $r1 \neq r2 \neq r3 \neq i \in \{1, \cdots, N_p\}(\mathbf{z}^{old} = \mathbf{z}_i^p)$ or $r1 \neq r2 \neq r3 \in \{1, \cdots, N_p\}(\mathbf{z}^{old} = \mathbf{z}^g)$ are three distinct random integers, F is a positive real number that represents the step length which controls the amplification of the difference vector $(\mathbf{z}_{r2}^p - \mathbf{z}_{r3}^p)$, $j \in \{1, \cdots, n\}$, r_{cj} is the uniformly distributed random number corresponding to the jth variable, $P_m \in [0, 1]$ is the mutation probability which is determined by the user, $r_c \in \{1, \cdots, n\}$ is a randomly chosen dimension index which ensures that at least one variable of $\mathbf{z}^{old}$ is mutated.

2.3 Regeneration Strategy

In the process of optimization, if $\mathbf{z}_i(t) = \mathbf{z}_i^p(t) = \mathbf{z}^g(t)$, $\mathbf{z}_i(t+1)$ will be equal to $\mathbf{z}_i(t)$. Then, particle i will stop evolving and we say that it has entered into

stagnation state. In order to prevent this phenomenon and improve the probability of the algorithm to find the global optimum, the stagnated particle would be regenerated randomly in the search space by the following rule:

$$z_{i,j}(t) = z_{\min,j} + r_s(z_{\max,j} - z_{\min,j}) \tag{6}$$

where r_s is the random number uniformly generated in the range of [0,1]. By this method, the stagnated particle will continue to evolve. Then, the diversity of the algorithm could be improved and the population will have more probability to find the global optimum.

2.4 Feasibility-Based Comparison Method

In this paper, the feasibility-based method is employed to compare the performances of different solutions. In this method, the objective function and constraints are treated separately, and the constraint violation precedes the objective function. The objective value and the constraints violation value of a potential solution $\mathbf{z}$ can be respectively represented by $f(\mathbf{z})$ and $\varphi(\mathbf{z})$.

Supposing that $f(\mathbf{z}_1)(f(\mathbf{z}_2))$ and $\varphi(\mathbf{z}_1)(\varphi(\mathbf{z}_2))$ are the objective values and constraint violation values for two compared solutions, the comparison rules for minimization problems are

- If $\varphi(\mathbf{z}_1) = \varphi(\mathbf{z}_2)$ and $f(\mathbf{z}_1) < f(\mathbf{z}_2)$, $\mathbf{z}_1$ is better than $\mathbf{z}_2$.
- If $\varphi(\mathbf{z}_1) < \varphi(\mathbf{z}_2)$, $\mathbf{z}_1$ is better than $\mathbf{z}_2$.

Thus, the objective function needs to be evaluated only when the constraint violation value of solution $\mathbf{z}_1$ is equal to or lower than that of solution $\mathbf{z}_2$. The evaluation of the objective function can often be omitted, and the computation time could be greatly cut down.

The feasibility-based comparison method has the following features:

- If both solutions are feasible, the one with lower objective value is better.
- Feasible solution is better than infeasible one.
- If both solutions are infeasible and have different constraint violation values, the one with lower constraint violation value is better.
- If both solutions are infeasible and have equal constraint violation values, the one with lower objective value is better.

Thus, the feasible solution will be found quickly. The feasible one with higher objective value will win in the end.

2.5 The Improved Gaussian Particle Swarm Optimization Algorithm

The framework of the improved GPSO based on DE mutation, the regeneration strategy and the feasibility-based comparison method (DGPSO) is described as Fig. 1.

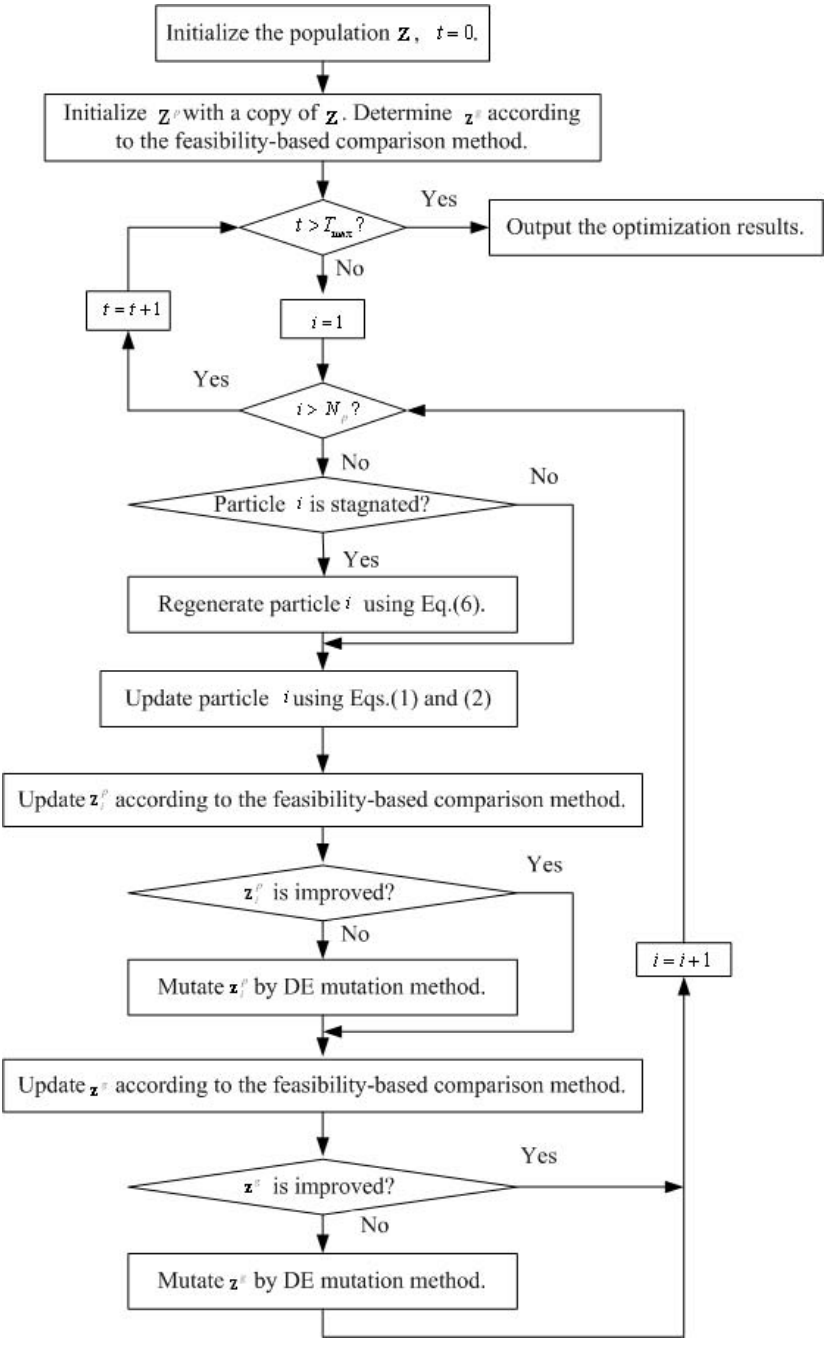


Fig. 1. The framework of DGPSO

3 Simulation Results for Engineering Optimization Problems

In this paper, three constrained engineering problems are tested to demonstrate the effectiveness of the proposed DGPSO. These problems are Himmelblau's problem, welded beam design problem and tension string design problem. The description of these problems were given in [1] [6] [9] and omitted here due to limited space. They have been used as benchmark problems for several population-based methods. Recently, Himmelblau's problem has been solved by ε constrained particle swarm optimization (εPSO) [4], the hybrid algorithm of εPSO and εGA (εPSO-GA) [4], and εPSO with adaptive velocity limit control (adaptive εPSO) [5]. The welded beam design problem has been solved by εPSO[4], εPSO-GA [4], adaptive εPSO [5], the hybrid PSO with a feasibility-based rule and simulation annealing (HPSO) [6], the co-evolutionary DE (CDE) [8], the co-evolutionary PSO (CPSO) [9], and the hybrid PSO with DE (PSO-DE) [10]. The tension string design problem has been solved by HPSO [6], CDE [8], CPSO [9], and PSO-DE [10].

In the present study, DGPSO is used to solve the above engineering problems. In the simulation, the population size is 30. The step length F is 0.8 [17], and the mutation rate P_m is 0.8. The maximum iteration $T_{\max}$ is 2000. 30 independent runs are performed for each problem.

Table 1~Table 3 are the comparison of the optimal results. The columns label Best, Average, Worst and S.D. are the best value, the average value, the worst value and the standard deviation for the best particle of 30 runs.

Table 1. Comparison of results for Himmelblau's problem

Algorithm	Best	Average	Worst	S.D.
εPSO	-31011.9988	-30947.3262	-30762.8890	55.8631
εPSO-GA	-31016.8002	-30952.2531	-30855.6951	38.8166
adaptive εPSO	-31022.3463	-30990.3279	-30873.6902	55.8631
GPSO	-30960.692	-30696.131	-30312.660	158.9262
DGPSO	**-31025.560**	**-31025.560**	**-31025.560**	**0**

Table 2. Comparison of results for welded beam design problem

Algorithm	Best	Average	Worst	S.D.
εPSO	1.7258	1.8073	2.1427	0.12
εPSO-GA	1.7268	1.7635	1.9173	0.0463
adaptive εPSO	1.7249	1.7545	1.8558	0.0370
HPSO	1.724852	1.749040	1.814295	0.040049
CDE	1.733461	1.768158	1.824105	0.022194
CPSO	1.724860	1.725015	1.725254	1.1468e-004
PSO-DE	1.7248531	1.7248579	1.7248881	4.1×10^{-6}
GPSO	1.7539490	2.5354191	3.6871943	0.4766
DGPSO	**1.7248523**	**1.7248523**	**1.7248523**	**9.0649**$\times10^{-16}$

Table 3. Comparison of results for tension string design problem

Algorithm	Best	Average	Worst	S.D.
HPSO	0.0126652	0.0127072	0.0127191	1.5824×10^{-5}
CDE	0.0126702	0.012703	0.012790	2.7×10^{-5}
CPSO	0.012681	0.012931	0.013308	1.5667×10^{-4}
PSO-DE	0.0126652	0.0126653	0.0126653	1.2×10^{-8}
GPSO	0.0126822	0.0146816	0.0177344	1.7484×10^{-3}
DGPSO	**0.0126652**	**0.0126652**	**0.0126652**	**1.6457**$\times10^{-18}$

Based on the above comparisons, DGPSO showed the best performance for these three engineering optimization problems. It finds not only the best optima for all three problems but also very low standard deviation for each problem. Among the 30 independent runs, the average number of objective function evaluations is 37149 with 119742 efficient particles movements for Himmelblau's problem, 49285 with 117497 for welded beam design problem, and 49285 with 117497 for tension string design problem. It shows that DGPSO is an efficient method for locating the global optimum.

4 Conclusions and Future Works

In this paper, an improved Gaussian particle swarm optimization algorithm is proposed for constrained optimization problems. In order to demonstrate the effectiveness of the proposed method, it is applied to solve three engineering problems. Simulation results illustrate that the proposed method is more accurate and robust than the compared algorithms. The number of objective function evaluations by the proposed method is less than half of that of efficient particles movements. The computational execute time is greatly cut down, which is very important for complicated optimization problems.

In the future, DGPSO will be applied to the wind farm micro-siting problem in a spatially- continuous manner. Selection of the mutation probability will be an important work for this complicated engineering problem.

Acknowledgment

This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2007AA05Z426 and the National Natural Science Foundation of China under Grant No. 61075064.

References

1. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)

2. Eberhart, R.C., Shi, Y.: Particle swarm optimization: developments, applications and resources. In: Congress on Evolutionary Computation, Kennedy (2001)
3. Schutte, J.F., Groenwold, A.A.: A study of global optimization using particle swarms. Global Optimization 31(1), 93–108 (2005)
4. Takahama, T., Sakai, S., Iwane, N.: Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In: Zhang, S., Jarvis, R.A. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 389–400. Springer, Heidelberg (2005)
5. Takahama, T., Sakai, S., Iwane, N.: Solving constrained optimization problems by the ϵ constrained particle swarm optimizer with adaptive velocity limit control. In: IEEE Conference on Cybernetics and Intelligent Systems, pp. 1–7 (2006)
6. He, Q., Wang, L.: A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation 186, 1407–1422 (2007)
7. Sun, C., Zeng, J., Pan, J.: An improved particle swarm optimization with feasibility-based rules for constrained optimization problems. In: Chien, B.-C., Hong, T.-P., Chen, S.-M., Ali, M. (eds.) IEA/AIE 2009. LNCS, vol. 5579, pp. 202–211. Springer, Heidelberg (2009)
8. Huang, F., Wang, L., He, Q.: An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and Computation 186, 340–356 (2007)
9. He, Q., Wang, L., Huang, F.: Nonlinear constrained optimization by enhanced co-evolutionary PSO. In: IEEE World Congress on Computational Intelligence, pp. 83–89 (2008)
10. Liu, H., Cai, Z., Wang, Y.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Applied Soft Computation 10, 629–640 (2010)
11. Krohling, R.A., Coelho, L.D.S.: Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 36(6), 1407–1416 (2006)
12. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, Perth, Australia, vol. 4(6), pp. 1942–1948 (1995)
13. Krohling, R.A.: Gaussian swarm: a novel particle swarm optimization algorithm. In: IEEE Conference on Cybernetics and Intelligence Systems, Singapore, pp. 372–376 (2004)
14. Krohling, R.A., Hoffmann, F., Coelho, L.D.S.: Co-evolutionary particle swarm optimization for min-max problems using Gaussian distribution. In: Congress on Evolutionary Computation, vol. 1, pp. 959–964 (2004)
15. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Global Optimization 11, 341–359 (1997)
16. Das, S., Suganthan, P.N.: Differential evolution - a survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation (2010)
17. Soliman, O.S., Bui, L.T., Abbass, H.A.: The effect of a stochastic step length on the performance of the differential evolution algorithm. In: IEEE Congress on Evolutionary Computation, pp. 2850–2857 (2007)

Evolving Neural Networks: A Comparison between Differential Evolution and Particle Swarm Optimization

Beatriz A. Garro[1], Humberto Sossa[1], and Roberto A. Vázquez[2]

[1] Centro de Investigación en Computación – IPN
Av. Juan de Dios Bátiz, esquina con Miguel de Othón de Mendizábal
Ciudad de México, 07738, México
[2] Intelligent Systems Group, Facultad de Ingeniería – Universidad La Salle
Benjamín Franklin 47 Col. Condesa CP 06140 México, D.F
bgarrol@ipn.mx, hsossa@cic.ipn.mx,
ravem@lasallistas.org.mx

Abstract. Due to their efficiency and adaptability, bio-inspired algorithms have shown their usefulness in a wide range of different non-linear optimization problems. In this paper, we compare two ways of training an artificial neural network (ANN): Particle Swarm Optimization (PSO) and Differential Evolution (DE) algorithms. The main contribution of this paper is to show which of these two algorithms provides the best accuracy during the learning phase of an ANN. First of all, we explain how the ANN training phase could be seen as an optimization problem. Then, we explain how PSO and DE could be applied to find the best synaptic weights of the ANN. Finally, we perform a comparison between PSO and DE approaches when used to train an ANN applied to different non-linear problems.

1 Introduction

A feed-forward artificial neural network (ANN) is a powerful tool widely used in the field of pattern recognition and time series analysis. However, despite their power in some practical problems, ANNs cannot reach an optimum performance in several non-linear problems. This fact is caused because the parameters, used during learning phase such as learning rate, momentum, among others, do not allow computing the best set of synaptic weights. In the field of the evolutionary computation, there are many algorithms that allow obtaining an optimum (or more) of a specific problem. One of these techniques is the Differential Evolution (DE) algorithm, based on the classical steps of the Evolutionary Computation. DE performs mutation based on the distribution of the solutions in the current population. In this way, search directions and possible step-sizes depend on the location of the individuals selected to calculate the mutation values.

On the other hand, the Particle Swarm Optimization (PSO) algorithm is inspired by observations of social interaction. PSO operates on a population of *particles* (that are the individuals), evolving them over a number of iterations with the goal of finding a solution to an optimization function. This metaphor searches an optimum solution based on the self and social experience of the best particle of the population.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 447–454, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Several works that use evolutionary strategies for training ANNs have been reported in the literature. For example, in [1], the authors combine PSO and ANNs for function approximation. Another application presented in [2] is a PSO-based neural network in the analysis of outcomes of construction claims in Hong Kong. Other examples can be found in [3], [4], [5], [6], [7] and [8].

DE algorithm has been less used in this kind of work. For example, Ilonen et al [9], propose the learning of an ANN by finding the synaptic weighs through the classical DE. Other examples can be found in [10] and [11]. In [12], it shows a large literature review where the evolutionary algorithms are used to evolve the synaptic weights.

In this paper, we compare the accuracy of PSO and DE algorithms during training the synaptic weights of ANN when applied to solve different classification problems. All of this, in order to determine which algorithm is better in the task of adjusting the synaptic weights of an ANN. This comparative is important because these two algorithms have got a different scheme each one. The way to manipulate the data for doing a new population is the key to find a good solution. The task of training a set of synaptic weights for an ANN is an essential aim and using these two algorithms when observing the ANN behavior. The aim of this paper is to explain how the neural network training phase could be seen as an optimization problem and how PSO and DE could be applied to find the best synaptic weights of the ANN. In the next section, we present a basic definition of an ANN. In section 3 basic PSO algorithm functioning is explained, while section 4 is devoted for DE algorithm. In section 5, we explain how PSO and DE could be applied to find the best synaptic weights of the ANN. In section 6 we perform a comparison between several ANNs trained by means of PSO and DE algorithms, when applied to solve different non-linear problems. In section 7, we finally give the conclusions of those experimental results.

2 Basics on Feed-Forward Neural Networks

A basic description of an ANN can be: a massively parallel-distributed processor made up from simple processing units. This type of processing unit performs in two stages: weighted summation and some type of nonlinear function. Each value of an input pattern $\mathbf{A} \in \mathbb{R}^N$ is associated with its weight value $\mathbf{W} \in \mathbb{R}^N$, which is normally between 0 and 1. Furthermore, the summation function often takes an extra input value θ with weight value of 1 to represent threshold or *bias* of a neuron. The summation function will be then performed as,

$$y = f\left(\sum_{i=1}^{N} a_i w_i + \theta\right) \tag{1}$$

The sum-of-product value is then passed into the second stage to perform the activation function $f(x)$ which generates the output from the neuron and determines the behavior of the neural model. In a multilayer structure the input nodes, which received the pattern $\mathbf{x} \in \mathbb{R}^N$, the units in the first hidden layer, then the outputs from the first hidden layer are passed to the next layer, and so on until they reach the output layer and produce an approximation of the desired output $\mathbf{y} \in \mathbb{R}^M$.

Basically, learning is a process by which the free parameters (i.e., synaptic weights $\mathbf{W}$ and bias levels θ) of an ANN are adapted through a continuous process using a labeled set of training data made up of p input-output samples:

$$\mathbf{T}^{\xi} = \left\{ \left(\mathbf{x}^{\xi} \in \mathbb{R}^{N}, \mathbf{d}^{\xi} \in \mathbb{R}^{M} \right) \right\} \forall \xi = 1, \ldots, p \tag{2}$$

where $\mathbf{x}$ is the input pattern and $\mathbf{d}$ the desired response.

Given the training sample $\mathbf{T}^{\xi}$, compute the free parameters of the neural network so that the actual output $\mathbf{y}^{\xi}$ of the neural network due to $\mathbf{x}^{\xi}$ is close enough to $\mathbf{d}^{\xi}$ for all ξ in a statistical sense. In this sense, we might use the mean-square error given in eq. 3 as the objective function to be minimized:

$$e = \frac{1}{p \cdot M} \sum_{\xi=1}^{p} \sum_{i=1}^{M} \left(d_i^{\xi} - y_i^{\xi} \right)^2 \tag{3}$$

3 Basics on Particle Swarm Optimization

PSO algorithm is a method for the optimization of continuous non-linear functions proposed by James Kennedy and Russell C. Eberhart. This algorithm is inspired on observations of social and collective behavior as well as fish schooling or bird flocking [13]. For example, a population or a flock could be considered as a cumulus of particles i where each particle represents the position $\mathbf{x}_i$ of a particle in a multidimensional space. These particles (individuals) also represent a possible solution of a specific function optimization. According to a velocity function $\mathbf{v}_i$ which takes into account the best position of a particle in a population $\mathbf{p}_g$ (i.e., social component) as well as the own best position of the particle $\mathbf{p}_i$ (i.e., cognitive component) the particles will move each iteration to a different position until they reach an optimum position. At each time step t, the velocity of a particle i is updated using

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 r_1 \left(\mathbf{p}_i(t) - \mathbf{x}_i(t) \right) + c_2 r_2 \left(\mathbf{p}_g(t) - \mathbf{x}_i(t) \right) \tag{4}$$

where ω is the inertia weight and typically setup to vary linearly from 1 to near 0 during the course of an iteration run; c_1 and c_2 are acceleration coefficients; $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$ are uniformly distributed random numbers in the range $(0,1)$. The velocity $\mathbf{v}_i$ is limited to the range $[v_{\min}, v_{\max}]$. Updating the velocity in this way enables the particle i to search around its individual best position $\mathbf{p}_i$, and the global best position $\mathbf{p}_g$. Based on the updated velocities, the new position of the particle i is computed using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \tag{5}$$

Finally, this optimum position will be the solution which maximizes or minimizes an objective function.

The basic PSO algorithm for the real version could be performed as follows:

Given a population of $\mathbf{x}_i \in \mathbb{R}^D, i = 1,\ldots,M$ individuals

1) Initialize the population at random
2) Until a stop criterion is reached:
   a) For each individual $\mathbf{x}_i$ evaluate their fitness.
   b) For each individual i, update its best position $\mathbf{p}_i$.
   c) From all individual i, update the best individual $\mathbf{p}_g$.
   d) For each individual i, compute the velocity update equation $\mathbf{v}_i(t+1)$ and then compute the current position $\mathbf{x}_i(t+1)$.

4 Basics on Differential Evolution

In 1995 an adaptive and efficient scheme emerged: Differential Evolution algorithm, proposed by Kenneth Price and Rainer Storn, useful for Global Optimization over continuous spaces [14]. With this precedent, it was opened a new optimization technique in Evolutionary Computation. Due to its exploration capacity over a search space of a given problem, the DE algorithm avoids staying in a local optimum. It has few parameters and it converges to the optimum faster than others evolutionary techniques (the solution's representation is given by vectors of real numbers). All these characteristics convert DE into an excellent optimization algorithm of a complex, non-differential and non-continuous problems.

The pseudo code of "DE/rand/1/bin" is shown in the next algorithm adapted from [15].

1. Randomly select two vectors from the current generation.
2. Use these two vectors to compute a difference vector.
3. Multiply the difference vector by weighting factor F.
4. Form the new trial vector by adding the weighted difference vector to a third vector randomly selected from the current population.

5 Evolving the Synaptic Weights of an ANN Using PSO and DE

In this section, we describe how given a set of patterns $\mathbf{T}$, the synaptic weights of an ANN can be automatically adjusted by means of a basic PSO and DE. The architecture of the ANN has to be previously defined because we only evolve the synaptic weights and the architecture must be static. The synapses weights of the ANN are codified based on a vector that represents a graph $\mathbf{x}$. The vector is a solution composed with the set of synaptic weights $w_{i,j}^k$ between neuron i and neuron j that belongs to the layer k, until a maximum number of neurons MNN. This vector represents an individual (particle or solution) of the population that will be evolved by PSO. The individual (see Fig. 1) has a set of values that change with respect to time and each

individual is evaluated in the fitness function in order to calculate the minimum square error generated by the ANN which maximizes the performance. The individual whose weights provoke the minimum value of the MSE will be the best solution for the trained ANN. This individual's representation is used by the two algorithms: PSO and DE.

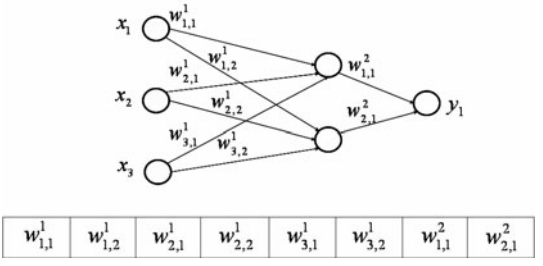


Fig. 1. Representation of an individual composed of a set of synaptic weights

Due to we are working with predefined feed-forward architectures, the fitness function which measures the performance of an individual is given by eq. 3 where the output y_i of the ANN is computed by means of equation 1.

6 Experimental Results

In order to evaluate the accuracy of the PSO and DE algorithms, several experiments were performed. Three well-known data-sets were taken from the UCI machine learning benchmark repository [16] to test the algorithms: iris plant, wine and breast cancer datasets. Also a real object recognition problem composed of 100 images of five different objects was used.

All datasets were randomly partitioned into two sets: 50% for the training data and 50% for the testing data. The input features of all data set were rescaled in a range between $[0,1]$ and the outputs were encoded by a winner-take-all representation. There are two conditions for the algorithm to stop: when it achieves the total number of generations, and when the algorithm achieves a $MSE = 1\times10^{-10}$. This last condition can be replaced using a validation set.

Before starting with the experiments, we defined the parameters of the three algorithms. For the case of the basic PSO algorithm, the population size M was set to 50, the number of generations was set to 4000, the initial position of the particles was in the range $[-40,40]$, velocity range $[-2,2]$, $\omega = 0.729$, $c_1 = c_2 = 2$. For the case of the basic DE algorithm, the population size $NP = 50$, number of generations was set to 4000, the population was initialized in the range $[-40,40]$, $CR = 0.9$, and $F = rand[0.3,0.7]$.

These parameters were set to the same value for all the experiments according to the literature (ω, c_1, c_2, F and CR). The others were defined by means of our criteria

and considering the limits of the numeric representations in the computer. The velocity range is different from 0 and not so big, in order to obtain possible directions without big displacements that can affect particle's movement. Before starting to evolve the synaptic weights, it was necessary to determine the architecture of the ANN. In this case, we decided to use an ANN with three layers: the input, the output and one hidden layer compose of five neurons.

Ten experiments for each dataset were performed to measure the accuracy of the proposed method. Figs. 2 and 3 show the percentage of recognition achieved with the ANN applied to solve the four problems by means of PSO and DE. The two algorithms found the best sets of synaptic weights that minimized the MSE of the ANN. After that, the classification error (CER) was calculated.

Fig. 2 shows the accuracy of the ANN trained with the PSO algorithm. As it can be observed from this figure, the best recognition percentages were achieved for the Iris plant and for the breast cancer data set, see Fig. 2 (a) and (c) respectively. For the case of the object recognition problem using PSO, the percentage of recognition was not as good as desired (Fig. 2 (d)). Furthermore, for the case of the wine data set, the percentage of recognition achieved by the ANN was the worst.

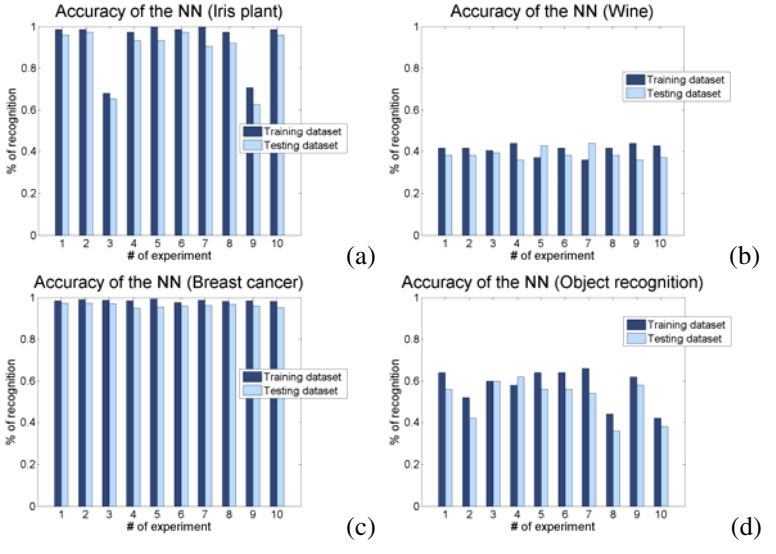


Fig. 2. Accuracy of the ANN using PSO algorithm. (a)Iris plant data set. (b)Wine data set. (c)Breast cancer data set. (d)Real object recognition data set.

Fig. 3 shows the accuracy of the ANN trained with the DE algorithm. Using this algorithm we can appreciate that the percentage of recognition for all the data sets was much better than the results obtained by PSO algorithm. Besides this, for the real object recognition problem, see Fig. 3(d), DE algorithm could obtain a percentage of recognition over the 90% in comparison with other results (Fig.3 (a), (b) and (c)). In addition to this, from Table 1 the reader can appreciate the average classification error (CER) for all experimental results just as the standard deviation for training phase and

testing phase. The results show that the best technique for evolving the synaptic weights of an ANN was the DE algorithm. This could be due to PSO algorithm has more parameters to tune than DE algorithm.

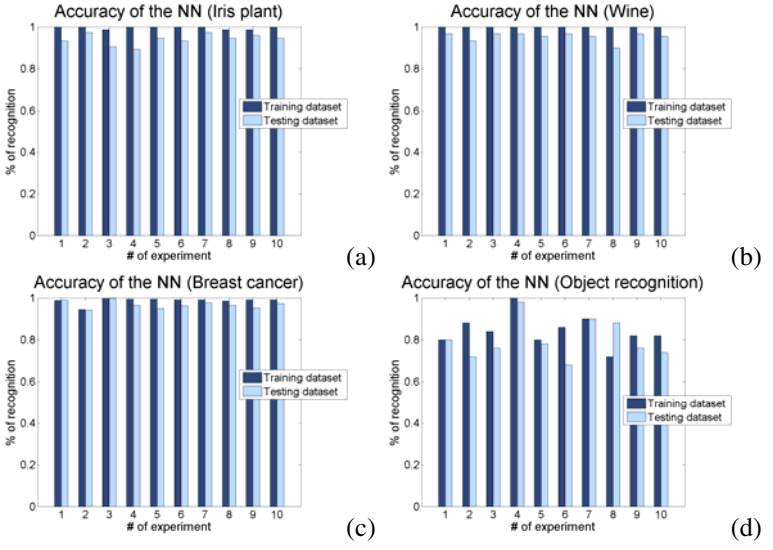


Fig. 3. Accuracy of the ANN using DE algorithm. (a)Iris plant data set. (b)Wine data set. (c)Breast cancer data set. (d)Real object recognition data set.

Table 1. Average and standard deviation applying the CER

Data base	PSO algorithm		DE algorithm	
	Tr. Er.	Te. Er.	Tr. Er.	Te. Er.
Iris plant	0.072 ± 0.124	0.116 ± 0.130	0.004 ± 0.006	0.058± 0.026
Wine	0.589 ± 0.026	0.612 ± 0.026	0 ± 0	0.047 ± 0.021
Breast Cancer	0.014 ± 0.026	0.037 ± 0.008	0.013 ± 0.015	0.032 ± 0.017
Object Recognition	0.424 ± 0.086	0.482 ± 0.094	0.156 ± 0.074	0.2 ± 0.092

Tr. Er = Training Error, Te. Er. = Testing Error.

7 Conclusions

In this paper, we compared two powerful bio-inspired algorithms in order to determine which one is more suitable to train an ANN: PSO and DE. This is very important because the training of an ANN is one of the keys issues to obtain a good generalization, and it is necessary to know the behavior of the evolutionary algorithms in the basic design of an ANN.

We explained in detail how the ANN's training phase could be seen as an optimization problem. Then, it was explained how PSO and DE could be applied to find the optimal synaptic weights of the neural network. Finally, we performed a comparison between a neural network trained with the PSO and DE algorithms, when applied to solve different non-linear pattern classification problems.

Through several experiments, we observed that DE algorithm was better in searching the best set of synaptic weights. This does not mean that PSO is not useful in this task, but it requires to determine the best set of parameters that improve its performance. For future woks, we need to test the experimentation applying PSO algorithm with neighborhoods due to this improvement avoid to be trapped in a local minimums.

Acknowledgements. H. Sossa thanks SIP-IPN, COTEPABE and DAAD-PROALMEX for economical support. Authors also thank the European Union, the European Commission and CONACYT for the economical support. This paper has been prepared by economical support of the European Commission under grant FONCICYT 93829. The content of this paper is an exclusive responsibility of the CIC-IPN and it cannot be considered that it reflects the position of the European Union.

References

[1] Tejen, S., Jyunwei, J., Chengchih, H.: A Hybrid Artificial Neural Networks and Particle Swarm Optimization for Function Approximation. International Journal of Innovative Computing, Information and Control ICIC 4, 2363–2374 (2008)
[2] Chau, K.W.: Application of a PSO-based neural network in analysis of outcomes of construction claims. Automation in Construction 16, 642–646 (2007)
[3] Chatterjee, A., et al.: A Particle Swarm Optimized Fuzzy-Neural Network for Voice Controlled Robot Systems. IEEE Trans. on Ind. Elec. 52, 1478–1489 (2005)
[4] Wang, Z., et al.: Particle Swarm Optimization and Neural Network Application for QSAR. In: Proceedings 18th Parallel and Distributed Processing Symposium (2004)
[5] Zhao, L., Yang, Y.: PSO-Based Single Multiplicative Neuron Model for Time Series Prediction. Expert Systems with Applications 36, 2805–2812 (2009)
[6] Da, Y., Ge, X.R.: An improved PSO–based ANN with simulated annealing technique. Neurocomput. Lett. 63, 527–533 (2005)
[7] Yu, J., et al.: An Improved Particle Swarm Optimization for Evolving Feedforward Artificial Neural Networks. Neural Processing Letters 26, 217–231 (2007)
[8] Mohaghegi, S., et al.: A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with STATCOM. In: Proceedings 2005 IEEE Swarm Intelligence Symposium on SIS 2005, pp. 381–384 (2005)
[9] Ilonen, J., Kamarainen, J.-K., Lampinen, J.: Differential Evolution Training Algorithm for Feed-Forward Neural Networks. Neural Processing Letters 17(1), 93–105 (2003)
[10] Castillo, P.A., Carpio, J., Merelo, J.J., Prieto, A., Rivas, V.: Evolving multilayer perceptrons. Neural Process. Lett. 12, 115–127 (2000)
[11] Rivero, D., Rabuñal, J.R., Dorado, J., Pazos, A.: Automatic design of aNNs by means of GP for data mining tasks: Iris flower classification problem. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007. LNCS, vol. 4431, pp. 276–285. Springer, Heidelberg (2007)
[12] Yao, X.: Evolving Artificial Neural Networks. Proc. of IEEE 87(9), 1423–1446 (1999)
[13] Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
[14] Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, TR-95-012, ICSI (1995)
[15] Mezura-Montes, E., Velazquez-Reyes, J., Coello Coello, C.A.: A Comparative Study of Differential Evolution Variants for Global Optimization. In: GECCO (2006)
[16] Murphy, P.M., Aha, D.W.: UCI repository of machine learning databases. Dept. Inf. Comput. Sci., Univ. California, Irvine, CA (1994)

Identification of Hindmarsh-Rose Neuron Networks Using GEO Metaheuristic

Lihe Wang[1], Genke Yang[1], and Lam Fat Yeung[2]

[1] Department of Automation, Shanghai Jiao Tong University,
Key Laboratory of System Control and Information Processing,
Ministry of Education of China, 200240, Shanghai, China
[2] Department of Electronic Engineering, City University of Hong Kong,
Hong Kong, China
{lh.wang,gkyang}@sjtu.edu.cn

Abstract. In the last few years bio-inspired neural networks have interested an increasing number of researchers. In this paper, a novel approach is proposed to solve the problem of identifying the topology and parameters in Hindmarsh-Rose-neuron networks. The approach introduces generalized extremal optimization (GEO), a relatively new heuristic algorithm derived from co-evolution to solve the identification problem. Simulation results show that the proposed approach compares favorably with other heuristic algorithms based methods in existing literatures with smaller estimation errors. And it presents satisfying results even with noisy data.

Keywords: biological neural network (BNN), identification, generalized extremal optimization (GEO), co-evolution method.

1 Introduction

In recent years, biological neural network (BNN) has aroused increasing interests among researchers in various fields. One of the popular emulations of BNN gives birth to the well-known artificial neural network (ANN), with a mimic of neuron system consisted of individual neurons synaptically coupled, and its application in pattern recognition, artificial intelligence etc. is enormous. The complexity of BNN is governed by neural dynamics as well as network structure. For instance, the topology has an impact on network synchronization and learning dynamics [1], [2], [3], hence the identification of BNN is a challenging task that receives much attention. Among many neuron models present [4], this paper focuses on Hindmarsh-Rose (HR) neuron model for its popularity in researches [5].

Generally there are two major approaches under the framework of synchronization towards the identification of HR-neuron networks in the last few decades. In electrical coupling networks, adaptive observers are applied to identify the topology [6]. And in synaptic coupling networks, since the complexity of the problem lies in its non-linear nature of synaptic coupling dynamics, many heuristic-based optimization algorithms are applied. In previous literatures, genetic algorithm (GA), jumping gene GA (JGGA), simulated annealing (SA), and tabu search (TS) are used, where JGGA outperforms the other three with better estimation results [7], [8].

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 455–463, 2011.
© Springer-Verlag Berlin Heidelberg 2011

In this paper, a novel approach is proposed to solve the identification problem in synaptic coupling HR-neuron networks. Firstly, a decoupled formation of the problem is given. And then it is solved by generalized extremal optimization (GEO) [9]. GEO is a method derived from co-evolution [10] and is inspired by Bak-Sneppen model that emulates natural selection mechanism [11]. The experimental results show that the proposed approach achieves better performance than those in previous literatures.

The paper is organized as follows. In Section 2, the preliminaries and problem formation are introduced. In Section 3, the detailed application of GEO is explained. In Section 4, experimental results are presented. Finally, conclusions are given.

2 Problem Description

2.1 Preliminaries

A neuron is an electric-excitable cell which transmits information via electrical and chemical signaling. The signaling process takes place in synapses, where signals from the axon of one neuron pass to a dendrite of another. When membrane potential changes, action potential travels along the axon of a neuron. The HR neuron model [5] was originally proposed to modify the membrane potential of a neuron. Fig.1 shows an example of membrane potential dynamics of HR neuron model.

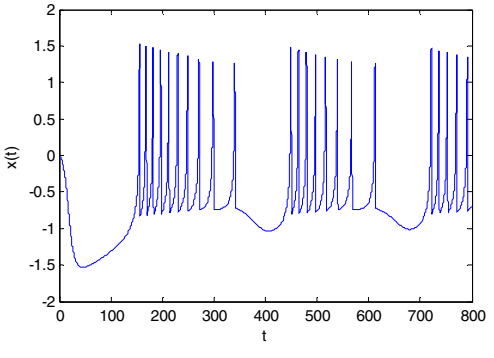


Fig. 1. Dynamics of membrane potential of HR-neuron model

Neurons coupled together to form neuron networks of different scales. In an N-neuron synaptic coupling network, the evolution dynamics of the i-th neuron is governed by a system of nonlinear ODEs [2], [7]:

$$\begin{cases} \dot{x}_i(t) = f_x(x_i, y_i, z_i) + g_s \sum_{j=1}^{N} h_{ij} \sigma_{V_s}(x_i) \gamma_{v,\theta_s}(x_j) \\ \dot{y}_i(t) = f_y(x_i, y_i, z_i) \\ \dot{z}_i(t) = f_z(x_i, y_i, z_i) \end{cases} \tag{1}$$

Where $x_i(t)$ is the variable for membrane potential, $y_i(t)$ and $z_i(t)$ are the spiking and bursting variables measuring the rate of transportation in fast and slow ion channels respectively. g_s is coupling strength, V_s , v and θ_s are free parameters modifying synaptic coupling. And h_{ij} are binary elements of connectivity matrix H , with $h_{ii}=0$; $h_{ij}=h_{ji}=1$ indicating a coupling between neuron i and j , and 0 otherwise. Explicit expressions of those functions in (1) are shown in (2):

$$\begin{aligned} f_x(x_i,y_i,z_i) &= a_i x_i^2 - x_i^3 - y_i - z_i \\ f_y(x_i,y_i,z_i) &= (a_i+\alpha_i)x_i^2 - y_i \\ f_z(x_i,y_i,z_i) &= \mu_i(b_i x_i + c_i - z_i) \\ \sigma_{V_s}(x_i) &= -(x_i - V_s) \\ \gamma_{v,\theta_s}(x_j) &= 1/(1+e^{-v(x_j-\theta_s)}) \end{aligned} \tag{2}$$

2.2 Problem Formation

As it is suggested [7], [8], assume that the time series of each neuron's membrane potential $\{x_i(t)\}$ are available, and all other parameters are known except for connectivity matrix $\{h_{ij}\}$ and $\{\alpha_i\}$ of each neuron. According to [6], [7], [8], a synchronization-based observer is adopted for network identification, where k is the feedback gain.

$$\begin{cases} \dot{\hat{x}}_i(t) = a_i\hat{x}_i^2 - \hat{x}_i^3 - \hat{y}_i - \hat{z}_i + g_s\sum_{j=1}^{N} h_{ij}^*\sigma_{V_s}(\hat{x}_i)\gamma_{v,\theta_s}(\hat{x}_j) - k(\hat{x}_i - x_i) \\ \dot{\hat{y}}_i(t) = (a_i+\alpha_i^*)\hat{x}_i^2 - \hat{y}_i \\ \dot{\hat{z}}_i(t) = \mu_i(b_i\hat{x}_i + c_i - \hat{z}_i) \end{cases} \tag{3}$$

Therefore, the identification can be formed into a minimization problem [7], [8]:

$$\min_{h_{ij}^*,\alpha_i^*} \sum_i \sum_t (x_i(t) - \hat{x}_i(t))^2 \tag{4}$$

If $\{\hat{x}_j\}$ is replaced by available data $\{x_j(t)\}$ and when objective function value reaches optima zero, it indicates $h_{ij}^* = h_{ij}$, $\alpha_i^* = \alpha_i$, and the following decoupled optimization problem for each neuron i is also minimized to zero.

$$\min_{h_{ij}^*,\alpha_i^*} \sum_t (x_i(t) - \hat{x}_i(t))^2 \tag{5}$$

In order to evaluate optimization results, define topology estimation error e_{top}, and parameter estimation error e_{par} as suggested in [7], [8]. When correct topology and parameters are obtained, both of the errors are minimized and equal to zero.

$$e_{top} = \sum_i \sum_j | h_{ij} - h_{ij}^* | \tag{6}$$

$$e_{par} = \sum_i | \alpha_i - \alpha_i^* | \tag{7}$$

3 Methodology

3.1 Solution Encoding

The population of coded solution of the GEO metaheuristic to solve the N-neuron network identification problem is composed of N binary strings, each with $m+N$ bits, where m bits is the differential encoding of α_i meeting accuracy requirements, and N bits is the encoding of h_{ij} with respect to neuron i, and h_{ii} is fixed 0 . Fig.2 shows an example of solution encoding.

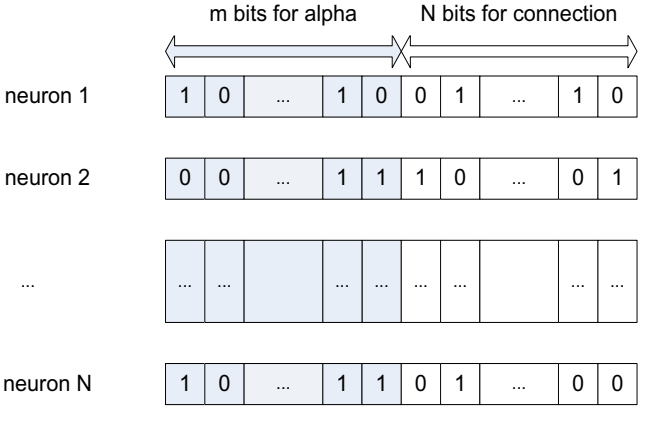


Fig. 2. Solution encoding

3.2 Identification Using GEO Metaheuristic

Extremal optimization (EO) is inspired by biological evolutionary Bak-Sneppen model [10], [12], which mimics natural selection by assigning each species a 'fitness' for its adaptability, and mutating the one with the lowest 'fitness' at each update. The extension of EO proposed by de Sousa and Ramos [9] is called Generalized Extremal

Optimization (GEO). Instead of always replacing the least-fit species, it generates a list of probabilities with power-law distribution, as the observed characteristics in self-organized criticality (SOC) in natural system [13], [14], to demonstrate the likelihood of mutation for each species. The power-law distribution has the following form.

$$P_i = m_i^{-\tau} \tag{8}$$

Where P_i is the mutation probability for i-th species (bit), m_i is the ranking of the corresponding species (bit) related to 'fitness', the higher the 'fitness', the less probability of mutation, and τ is a positive adjustable parameter. When $\tau \to \infty$, the algorithm performs a deterministic search. When $\tau \to 0$, the algorithm conducts a random search [15]. The GEO algorithm is simple for it has only one parameter τ to adjust, yet very effective, for it exhibits strong search ability due to its non-equilibrium nature [13]. A step description of applying GEO heuristic to the problem is listed below.

1. Initializing N binary strings each with $m+N$ bits randomly, where m bits should be the coding of α_i, and N is the number of neurons in the network.
2. Set the current configuration C and its objective function value V as the best, C_{best} and V_{best} respectively.
3. For each binary string S_i, do,
   - For every bit in the string, flip one bit at a time to acquire a new configuration C_{tempi} and assign its objective function value V_{tempi} to the bit according to (5), and flip it back as the original before flip anther.
   - Set fitness to each bit in the configuration C_i of string S_i with respect to its objective function values. Lower fitness indicates a relative loss in the corresponding objective function value mutating the bit in a minimization problem.
   - Perform sorting according to bit fitness, the lower the fitness the higher the rank, the bits with equal fitness are ranked randomly. Then flip the corresponding bit with probability given by (8), and generate a new configuration C_{newi} for string S_i and its objective function value V_{newi} according to (5).
   - Update $C_i = C_{newi}$, and $V_i = V_{newi}$.
   - If $V_{newi} < V_i$ for minimization problem, update $C_{besti} = C_{newi}$, $V_{besti} = V_{newi}$.
4. Consolidate C_i and V_i of each binary string S_i into C and V and update C_{best} and V_{best} accordingly.
5. If termination condition is not satisfied (e.g. maximum iteration is not reached), repeat step 3 and step 4.
6. Feasibility checks for output, if not feasible, discard it and go to step 1.
7. Output C_{best} and V_{best}.

4 Simulation

4.1 The Parameters

In the simulation, we use the same parameters and test network as suggested in [7], [8]. The parameters are: $a = 2.8$, $b = 9$, $c = 5$, $\alpha \in [0.35, 2.90]$, $\mu = 0.001$, $v = 10$, $\theta_s = -0.25$, $V_s = 2$, and $g_s = 0.34$. We set: $m = 8$, $N = 8$, and GEO algorithm parameters: $\tau = 1.5$, $maximum\,iteration = 120$. The 8-HR-neuron test network is shown in Fig.3.

4.2 The Results

In the simulation, the average results are given on the basis of 30 independent runs.

In Fig.4, e_{par} and e_{top} against iteration by 30 independent runs are shown.

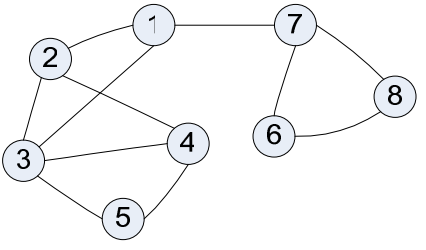


Fig. 3. An example of 8-HR-neuron network as test network

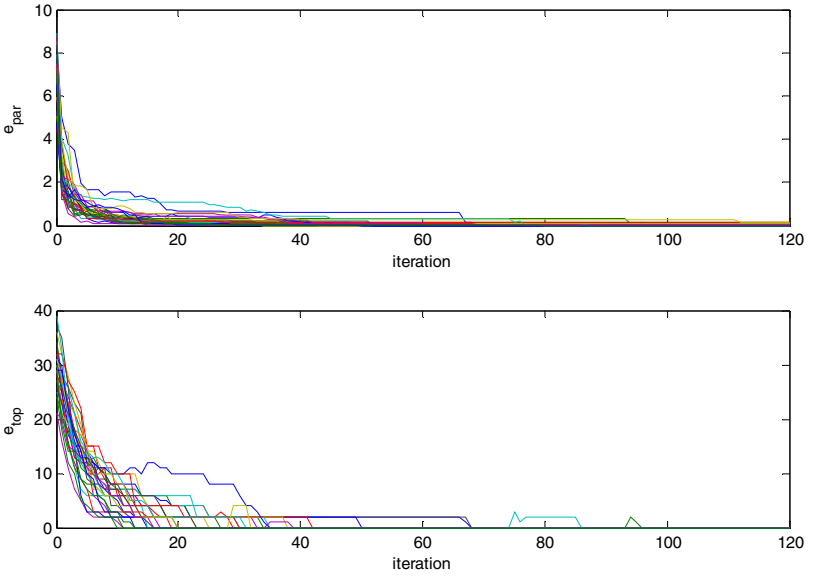


Fig. 4. Estimation errors e_{par} and e_{top} against iteration

Fig.4. shows that at the beginning, some estimation errors exist because the initial configuration is random. And then the estimation errors converge to a relatively small value. Also, the parameter estimation errors present a slightly faster convergence than topology estimation errors. Overall, both of the parameter and topology estimation errors decrease against iteration, and the topology estimation errors of the 30 runs all converge to 0, indicating that correct topology can be found. However, small estimation errors e_{par} may still occur in the final results, which are expected to be improved by increasing the number of iterations.

In order to fully evaluate the performance of the proposed GEO-based method, a comparison with other heuristic-algorithm-based approaches to the same problem is presented in Tab.1, where those algorithms include GA, JGGA, SA and TS as suggested in [8].

Table 1. The performance comparision (The performance data of GA, JGGA,SA and TS are obtained from [8])

Algorithms	Average e_{top}	Average e_{par}
GEO based method	0	0.029
GA	0	0.191
JGGA	0	0.164
SA	0	0.191
TS	0	0.171

The results shown in Tab. 1 demonstrate a competitive performance of the proposed method with smaller e_{par}, and all the methods can obtain the network topology correctly. The effectiveness of the proposed method is due to the combination of decoupled problem formation which reduces the searching space, and the strong search and hill-climbing ability of GEO algorithm. However, the GEO evaluate more candidate solutions, since it performs evaluation every time when assigns fitness to each bit in the configurations.

While in most real life cases, the observable data $\{x_i(t)\}$ are usually acquired with noises. The noisy-data model is shown in (9), where $x_i'(t)$ is the data without noise, $x_i(t)$ is the data acquired, and $\omega(t)$ is a zero-mean Gaussian White Noise.

$$x_i(t) = x_i'(t) + \omega(t) \tag{9}$$

The noisy-data model is used to evaluate the robustness of the proposed method. The performance of the method under different data noise levels are shown in Tab. 2.

As it is shown in Tab. 2, the average parameter estimation error increases, as the data noise level increases, this indicates that data noises have some influence on the identification results. Intuitively, noise may contaminate data, which makes it hard to recover information from it. However, the average parameter estimation errors are still relatively small, and the topology of the network can be obtained.

Table 2. The performance with noisy data

Signal to Noise Ratio (SNR)	Noise Level	Average e_{top}	Average e_{par}
100dB	~0%	0	0.029
26dB	5%	0	0.045
20dB	10%	0	0.117

5 Conclusions

In this paper, a novel approach of identifying topology and parameters in HR-neuron networks is proposed. The approach incorporates GEO, a relatively new evolutionary metaheuristic, to solve the identification problem. By comparing with other heuristic-based approaches in recent literatures, the experiments demonstrate that the proposed approach presents more accurate results, and it performs well even if the observational data have up to 10% noise levels. Since in real cases, data are usually acquired with noises, therefore the robustness of the approach may lead it to more practical BNN-identification applications.

Acknowledgments. The work of this paper was supported by National Natural Science Foundation of China, No.61074150, and No.60574063.

References

1. Belykh, I., Lange, E., Hasler, M.: Synchronization of Bursting Neurons: What Matters in the Network Topology. J. Phys. Rev. Lett. 94(18), 188101 (2005)
2. Checco, P., Righero, M., Biey, M., Kocarev, L.: Synchronization in Networks of Hindmarsh-Rose Neurons. J. IEEE Trans. on Circuits and Systems-II: Express Briefs 55(12), 1274–1278 (2008)
3. Streib, F.E.: Influence of the Neural Network Topology on the Learning Dynamics. J. Neurocomputing 69(10-12), 1170–1182 (2006)
4. Izhikevich, E.M.: Which Model to Use for Cortical Spiking Neurons. J. IEEE Trans. on Neural Network 15(5), 1063–1070 (2004)
5. Hindmarsh, J.L., Rose, R.M.: A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations. J. Proc. R. Soc. Lond. B 221, 87–102 (1984)
6. Mao, Y., Tang, W., Liu, Y., Kocarev, L.: Identification of Biological Neurons Using Adaptive Observers. J. Cognitive Processing 10 (supplement 1), 41–53 (2009)
7. Yin, J.J., Tang, W., Man, K.F.: Identification of Biological Neural Network Using Jumping Gene Genetic Algorithm. In: 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), pp. 693–697. IEEE Press, Taipei (2007)
8. Yin, J.J., Tang, W., Man, K.F.: A Comparison of Optimization Algorithms for Biological Neural Network Identification. J. IEEE Trans. on Industrial Electronics 57(3), 1127–1131 (2010)
9. de Sousa, F.L., Ramos, F.M., Galski, R.L., Muraoka, I.: Generalized External Optimization: A New Meta-Heuristic Inspired by a Model of Natural Evolution. J. Recent Developments in Biologically Inspired Computing 13(10), 41–60 (2004)

10. Boettcher, S., Percus, A.G.: Extremal Optimization: Methods derived from Co-Evolution. In: GECCO-1999: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 825–832. Morgan Kaufmann, San Francisco (1999)
11. Flyvbjerg, H., Sneppen, K., Bak, P.: Mean Field Theory for a Simple Model of Evolution. J. Phys. Rev. Lett. 71(24), 4087–4090 (1993)
12. Boettcher, S., Percus, A.G.: Nature's Way of Optimizing. J. Artificial Intelligence 119 (1-2), 275–286 (2000)
13. Lu, Y.Z., Chen, M.R., Chen, Y.W.: Studies on Extremal Optimization and Its Applications in Solving Real World Optimization Problems. In: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), pp. 162–168. IEEE Press, Honolulu (2007)
14. de Sousa, F.L., Vlassov, V., Ramos, F.M.: Generalized Extremal Optimization for solving complex optimal design problems. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 375–376. Springer, Heidelberg (2003)
15. Randall, M., Lewis, A.: Intensification Strategies for Extremal Optimisation. In: Deb, K., Bhattacharya, A., Chakraborti, N., Chakroborty, P., Das, S., Dutta, J., Gupta, S.K., Jain, A., Aggarwal, V., Branke, J., Louis, S.J., Tan, K.C. (eds.) SEAL 2010. LNCS, vol. 6457, pp. 115–124. Springer, Heidelberg (2010)

Delay-Dependent Stability Criterion for Neural Networks of Neutral-Type with Interval Time-Varying Delays and Nonlinear Perturbations

Guoquan Liu[1], Simon X. Yang[1,2], and Wei Fu[1]

[1] College of Automation, Chongqing University, 400044 Chongqing, China
guoquanliu1982@hotmail.com
[2] School of Engineering, University of Guelph, Guelph,Ontario, Canada N1G2W1

Abstract. In this paper, the delay-dependent stability problem for a class of neural networks of neutral-type with interval time-varying delays and nonlinear perturbations is investigated. A novel stability criterion is obtained in terms of linear matrix inequality (LMI) by employing a Lyapunov-Krasovskii functional. The proposed criteria can be checked easily by the LMI Control Toolbox in Matlab. In addition, two examples are given to show the effectiveness of the obtained result.

Keywords: Delay-dependent stability, Neural networks, Neutral-type, Lyapunov-Krasovskii functional, Nonlinear perturbations.

1 Introduction

Over the past decade, there has been a growing interest in the problem of exponential stability or asymptotic stability for neural networks with time delays. Time delays are often the sources of instability and encountered in various neural networks such as Hopfield neural networks, Cellular neural networks, Cohen-Grossberg neural networks. Therefore, stability criteria for neural networks with time delays have been attracted the attention of many researchers [1-12]. It is northing noting that the existing stability criteria can be classified into two categories: delay-independent stability and delay-dependent stability, and the latter are less conservative than the former. In recent years, there are some research issues about neural networks of neutral-type with (or neutral type neural networks) time delays in [13-17], where [15-17] considered the delay-dependent global stability results for neural networks of neutral-type with time delays.

On the other hand, the range of time-varying delay for neural networks of neutral-type considered in [18-19] is from zero to an upper bound. In the real world, a time-varying interval delay is often encountered, that is, the range of delay varies in an interval for which the lower bound is not restricted to zero. For example, based on the Lyapunov-Krasovkii functional theory and the LMI method, a novel delay-dependent global asymptotic stability criterion neutral type neural networks with interval time-varying delays was studied in [19]. Very recently, the problem of robust stability analysis for uncertain neural networks of neutral-type is considered by some researchers. Some useful stability conditions have been established in [20-21]. However, most

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 464–471, 2011.
© Springer-Verlag Berlin Heidelberg 2011

of the obtained results are based on restricting norm-bounded for parameter uncertainties. So far, the robust asymptotic stability problem has not been touched for neural networks of neutral-type with nonlinear perturbations, which is still open problem.

Based on the above discussion, the objective of this paper is to investigate the asymptotic stability criteria for neural networks of neutral-type with internal time-varying delays and nonlinear perturbations. For this purpose, the Lyapunov functional method is taken in our study. Finally, two numerical examples are given to demonstrate the applicability of the proposed stability criteria.

Notations: The following notations will be used throughout this paper. For a real square matrix X, the notation $X \geq 0$ (respectively, $X > 0$), means that X is positive semi-definite (respectively, positive definite); $\Re^n$ and $\Re^{n\times n}$ denote the n-dimensional Euclidean space and the set of all $n\times n$ real matrices, respectively; The superscripts "T" and "-1" stand for matrix transposition and matrix inverse, respectively; The shorthand $\mathrm{diag}\{X_1, X_2, ..., X_n\}$ denotes a block diagonal matrix with diagonal blocks being the matrices $X_1, X_2, ..., X_n$; The notation $*$ always denotes the symmetric block in one symmetric matrix.

2 Problem Statement

Consider the following neural networks of neutral-type with time-varying delays and nonlinear perturbations:

$$\begin{cases} \dot{x}(t) = -Ax(t) + Bf(x(t)) + Cf(x(t-\tau(t)) + D\dot{x}(t-\tau(t)) \\ \qquad + f_1(t, x(t)) + f_2(t, x(t-\tau(t))) + f_3(t, \dot{x}(t-\tau(t))), \\ x(\theta) = \vartheta(\theta), \dot{x}(\theta) = \delta(\theta), \forall \theta \in [-\tau_2, 0], \end{cases} \tag{1}$$

where $x(t) = [x_1(t), x_2(t), ..., x_n(t)]^T \in \Re^n$ is the state vector, $A, B, C, D \in \Re^{n\times n}$ are constant matrices with appropriate dimensions, $f(x(t)) = [f_1(x_1(t)), f_2(x_2(t)), ..., f_n(x_n(t))]^T \in \Re^n$ is the neuron activation function vector. $\tau(t)$ is a time-varying delay, and it is assumed to satisfy

$$0 \leq \tau_1 \leq \tau(t) \leq \tau_2, \dot{\tau}(t) \leq \tau_d < 1, \tag{2}$$

where τ_1, τ_2 and τ_d are constants. $\vartheta(\theta)$ and $\delta(\theta)$ are the initial condition functions that are continuously differentiable of $[-\tau_2, 0]$, $f_1(t, x(t)), f_2(t, x(t-\tau(t))), f_3(t, \dot{x}(t-\tau(t)))$, are unknown nonlinear perturbations. They satisfy that $f_1(t,0) = f_2(t,0) = f_3(t,0) = 0$, and

$$\begin{aligned} & f_1^T(t, x(t)) f_1(t, x(t)) \leq \alpha^2 x^T(t) x(t), \\ & f_2^T(t, x(t-\tau(t))) f_2(t, x(t-\tau(t))) \leq \beta^2 x^T(t-\tau(t)) x(t-\tau(t)), \\ & f_3^T(t, \dot{x}(t-\tau(t))) f_3(t, \dot{x}(t-\tau(t))) \leq \gamma^2 \dot{x}^T(t-\tau(t)) \dot{x}(t-\tau(t)), \end{aligned} \tag{3}$$

where $\alpha \geq 0, \beta \geq 0$ and $\gamma \geq 0$ are given constants, for the sake of simplicity, the following notations are adopted:

$$f_1 := f_1(t, x(t)), f_2 := f(t, x(t-\tau(t))), f_3 := f(t, \dot{x}(t-\tau(t))).$$

Then, system (1) can be re-written as

$$\begin{cases} \dot{x}(t) = -Ax(t) + Bf(x(t)) + Cf(x(t-\tau(t)) + D\dot{x}(t-\tau(t)) + f_1 + f_2 + f_3, \\ x(\theta) = \vartheta(\theta), \dot{x}(\theta) = \delta(\theta), \forall \theta \in [-\tau_2, 0]. \end{cases} \tag{4}$$

The following lemmas are useful in deriving the criterion.

Lemma 1. For any constant matrix $M > 0$, any scalars a and b with $a < b$, and a vector function $x(t):[a,b] \to \Re^n$ such that the integrals concerned as well defined, the following holds

$$\left[\int_a^b x(s)ds\right]^T M \left[\int_a^b x(s)ds\right] \le (b-a)\int_a^b x^T(s)Mx(s)ds. \tag{5}$$

Lemma 2(Schur complement). Given constant matrix M_1, M_2 and M_3 with appropriate dimensions, where $M_1^T = M_1$ and $M_2^T = M_2 > 0$, then $M_1 + M_3^T M_2^{-1} M_3 < 0$ if and only if

$$\begin{bmatrix} M_1 & M_3^T \\ * & -M_2 \end{bmatrix} < 0 \quad \text{or} \quad \begin{bmatrix} -M_2 & M_3 \\ * & M_1 \end{bmatrix} < 0. \tag{6}$$

3 Main Result and Its Proof

In this section, a novel delay-dependent stability criterion for the system (4) is derived in terms of LMI.

Theorem 1. For given scalars τ_1, τ_2 and τ_d, system (4) is asymptotically stable, if there exist matrices $P > 0$, $Q_j = Q_j^T > 0, j = 1,2,3$, $R_j = R_j^T > 0, j = 1,2,3$, $Z_i, i = 1,2$, a diagonal matrix $K > 0$, and positive scalars $\varepsilon_i > 0, i = 1,2,3$ such that the following LMI holds:

$$\Pi = \begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 \\ * & -\varphi_5 & 0 & 0 \\ * & * & -\varphi_6 & 0 \\ * & * & * & -\varphi_7 \end{bmatrix} < 0, \tag{7}$$

where

$$\varphi_1 = \begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} & 0 & 0 & \varphi_{1,5} & \varphi_{1,6} & \varphi_{1,7} & 0 & \varphi_{1,9} & \varphi_{1,10} & \varphi_{1,11} \\ * & \varphi_{2,2} & 0 & 0 & 0 & \varphi_{2,6} & \varphi_{2,7} & 0 & \varphi_{2,9} & \varphi_{2,10} & \varphi_{2,11} \\ * & * & \varphi_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & \varphi_{4,4} & \varphi_{4,5} & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & \varphi_{5,5} & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & \varphi_{6,6} & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * & \varphi_{7,7} & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * & * & \varphi_{8,8} & 0 & 0 & 0 \\ * & * & * & * & * & * & * & * & \varphi_{9,9} & 0 & 0 \\ * & * & * & * & * & * & * & * & * & \varphi_{10,10} & 0 \\ * & * & * & * & * & * & * & * & * & * & \varphi_{11,11} \end{bmatrix},$$

$$\varphi_2 = \begin{bmatrix} -AQ_3 & BQ_3 & 0 & 0 & 0 & CQ_3 & DQ_3 & 0 & 0 & 0 & 0 \end{bmatrix}^T,$$
$$\varphi_3 = \begin{bmatrix} -\tau_2 A & \tau_2 B & 0 & 0 & 0 & \tau_2 C & \tau_2 D & 0 & 0 & 0 & 0 \end{bmatrix}^T,$$
$$\varphi_4 = \begin{bmatrix} -(\tau_2 - \tau_1)A & (\tau_2 - \tau_1)B & 0 & 0 & 0 & (\tau_2 - \tau_1)C & (\tau_2 - \tau_1)D & 0 & 0 & 0 & 0 \end{bmatrix}^T,$$
$$\varphi_5 = Q_3, \varphi_6 = Z_1, \varphi_7 = Z_2,$$

with

$$\varphi_{1,1} = -PA - AP^T + Q_1 + Q_3 - Z_1 + (\tau_2 - \tau_1)^2 Z_2 + \varepsilon_1 \alpha^2, \varphi_{1,2} = PB - A^T K,$$
$$\varphi_{1,5} = Z_1, \varphi_{1,6} = PC, \varphi_{1,7} = PD, \varphi_{1,9} = P, \varphi_{1,10} = P, \varphi_{1,11} = P,$$
$$\varphi_{2,2} = KB + B^T K, \varphi_{2,6} = KC, \varphi_{2,7} = KD, \varphi_{2,9} = P, \varphi_{2,10} = P,$$
$$\varphi_{2,11} = P, \varphi_{3,3} = -(1-\tau_d)Q_1 + \varepsilon_2 \beta^2, \varphi_{4,4} = -R_1 - Z_2, \varphi_{4,5} = Z_2,$$
$$\varphi_{5,5} = -R_2 - Z_1 - Z_2, \varphi_{6,6} = -(1-\tau_d)Q_2, \varphi_{7,7} = -(1-\tau_d)Q_3 + \varepsilon_3 \gamma^2,$$
$$\varphi_{8,8} = -R_3, \varphi_{9,9} = -\varepsilon_1, \varphi_{10,10} = -\varepsilon_2, \varphi_{11,11} = -\varepsilon_3.$$

Proof: Consider the following lyapunov-krasoskill functional for system (4) as follows

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t), \tag{8}$$

where

$$V_1(t) = x^T(t)Px(t) + 2\sum_{i=1}^{n} k_i \int_0^{x_i(t)} f_i(s)ds,$$
$$V_2(t) = \int_{t-\tau(t)}^{t} x^T(s)Q_1 x(s)ds + \int_{t-\tau(t)}^{t} f^T(x(s))Q_2 f(x(s))ds + \int_{t-\tau(t)}^{t} \dot{x}^T(s)Q_3 \dot{x}(s)ds,$$
$$V_3(t) = \int_{t-\tau_1}^{t} x^T(s)R_1 x(s)ds + \int_{t-\tau_2}^{t} x^T(s)R_2 x(s)ds + (\tau_2 - \tau_1)\int_{-\tau_2}^{-\tau_1}\int_{t+\delta}^{t} x^T(s)R_3 x(s)dsd\delta,$$
$$V_4(t) = \tau_2 \int_{-\tau_2}^{0}\int_{t+\delta}^{t} \dot{x}^T(s)Z_1 \dot{x}(s)dsd\delta + (\tau_2 - \tau_1)\int_{-\tau_2}^{-\tau_1}\int_{t+\delta}^{t} \dot{x}^T(s)Z_2 \dot{x}(s)dsd\delta.$$

The time derivative of $V(t)$ along the trajectory of system (4) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t), \tag{9}$$

where

$$\begin{aligned} \dot{V}_1(t) &= 2x^T(t)P\left[-Ax(t) + Bf(x(t)) + Cf(x(t-\tau(t)) + D\dot{x}(t-\tau(t)) + f_1 + f_2 + f_3\right] \\ &\quad + 2f^T(x(t))K\left[-Ax(t) + Bf(x(t)) + Cf(x(t-\tau(t)) + D\dot{x}(t-\tau(t)) + f_1 + f_2 + f_3\right], \end{aligned} \tag{10}$$

$$\begin{aligned} \dot{V}_2(t) &= x^T(t)Q_1 x(t) - (1-\dot{\tau}(t))x^T(t-\tau(t))Q_1 x(t-\tau(t)) \\ &\quad + f^T(x(t))Q_2 f(x(t)) - (1-\dot{\tau}(t))f^T(x(t-\tau(t)))Q_2 f(x(t-\tau(t))) \\ &\quad + \dot{x}^T(t)Q_3 \dot{x}(t) - (1-\dot{\tau}(t))\dot{x}^T(t-\tau(t))Q_3 \dot{x}(t-\tau(t)) \\ &\le x^T(t)Z_1 x(t) - (1-\tau_d)x^T(t-\tau(t))Q_1 x(t-\tau(t)) \\ &\quad + f^T(x(t))Q_2 f(x(t)) - (1-\tau_d)f^T(x(t-\tau(t)))Q_2 f(x(t-\tau(t))) \\ &\quad + \dot{x}^T(t)Q_3 \dot{x}(t) - (1-\tau_d)\dot{x}^T(t-\tau(t))Q_3 \dot{x}(t-\tau(t)) \end{aligned} \tag{11}$$

$$\begin{aligned}\dot{V}_3(t) = {} & x^T(t)R_1x(t) - x^T(t-\tau_1)R_1x(t-\tau_1) + x^T(t)R_2x(t) - x^T(t-\tau_2)R_2x(t-\tau_2) \\ & + (\tau_2-\tau_1)^2 x^T(t)R_3x(t) - (\tau_2-\tau_1)\int_{t-\tau_2}^{t-\tau_1} x^T(s)R_3x(s)ds,\end{aligned} \tag{12}$$

$$\begin{aligned}\dot{V}_4(t) = {} & \tau_2^2\dot{x}^T(t)Z_1\dot{x}(t) - \tau_2\int_{t-\tau_2}^{t}\dot{x}^T(s)Z_1\dot{x}(s)ds + (\tau_2-\tau_1)^2\dot{x}^T(t)Z_2\dot{x}(t) \\ & - (\tau_2-\tau_1)\int_{t-\tau_2}^{t-\tau_1}\dot{x}^T(s)Z_2\dot{x}(s)ds.\end{aligned} \tag{13}$$

By using Lemma 1, one can get

$$-(\tau_2-\tau_1)\int_{t-\tau_2}^{t-\tau_1} x^T(s)R_3x(s)ds \le -\left[\int_{t-\tau_2}^{t-\tau_1} x(s)ds\right]^T R_3\left[\int_{t-\tau_2}^{t-\tau_1} x(s)ds\right], \tag{14}$$

$$\begin{aligned}-\tau_2\int_{t-\tau_2}^{t}\dot{x}^T(s)Z_1\dot{x}(s)ds &\le -\left[\int_{t-\tau_2}^{t}\dot{x}(s)ds\right]^T Z_1\left[\int_{t-\tau_2}^{t}\dot{x}(s)ds\right] \\ &= -[x(t)-x(t-\tau_2)]^T Z_1[x(t)-x(t-\tau_2)],\end{aligned} \tag{15}$$

$$\begin{aligned}-(\tau_2-\tau_1)\int_{t-\tau_2}^{t}\dot{x}^T(s)Z_2\dot{x}(s)ds &\le -\left[\int_{t-\tau_2}^{t-\tau_1}\dot{x}(s)ds\right]^T Z_2\left[\int_{t-\tau_2}^{t-\tau_1}\dot{x}(s)ds\right] \\ &= -[x(t-\tau_1)-x(t-\tau_2)]^T Z_2[x(t-\tau_1)-x(t-\tau_2)].\end{aligned} \tag{16}$$

From (3), one can obtain for any scalars $\varepsilon_1 > 0, \varepsilon_2 > 0$ and $\varepsilon_3 > 0$.

$$\begin{aligned}&0 \le \varepsilon_1\left[\alpha^2 x^T(t)x(t) - f_1^T f_1\right], \\ &0 \le \varepsilon_2\left[\beta^2 x^T(t-\tau(t))x(t-\tau(t)) - f_2^T f_2\right], \\ &0 \le \varepsilon_3\left[\gamma^2\dot{x}^T(t-\tau(t))\dot{x}(t-\tau(t)) - f_3^T f_3\right].\end{aligned} \tag{17}$$

Then, combining (10)-(17), one can obtain that

$$\dot{V}(t) \le \xi^T(t)\left[\varphi_1 + Q_3 + \tau_2^2 Z_1 + (\tau_2-\tau_1)^2 Z_2\right]\xi(t) = \xi^T(t)\Omega\xi(t), \tag{18}$$

where

$$\Omega = \varphi_1 + Q_3 + \tau_2^2 Z_1 + (\tau_2-\tau_1)^2 Z_2, \tag{19}$$

$$\varphi_1 = \begin{bmatrix}
\varphi_{1,1} & \varphi_{1,2} & 0 & 0 & \varphi_{1,5} & \varphi_{1,6} & \varphi_{1,7} & 0 & \varphi_{1,9} & \varphi_{1,10} & \varphi_{1,11} \\
* & \varphi_{2,2} & 0 & 0 & 0 & \varphi_{2,6} & \varphi_{2,7} & 0 & \varphi_{2,9} & \varphi_{2,10} & \varphi_{2,11} \\
* & * & \varphi_{3,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
* & * & * & \varphi_{4,4} & \varphi_{4,5} & 0 & 0 & 0 & 0 & 0 & 0 \\
* & * & * & * & \varphi_{5,5} & 0 & 0 & 0 & 0 & 0 & 0 \\
* & * & * & * & * & \varphi_{6,6} & 0 & 0 & 0 & 0 & 0 \\
* & * & * & * & * & * & \varphi_{7,7} & 0 & 0 & 0 & 0 \\
* & * & * & * & * & * & * & \varphi_{8,8} & 0 & 0 & 0 \\
* & * & * & * & * & * & * & * & \varphi_{9,9} & 0 & 0 \\
* & * & * & * & * & * & * & * & * & \varphi_{10,10} & 0 \\
* & * & * & * & * & * & * & * & * & * & \varphi_{11,11}
\end{bmatrix},$$

and

$$\xi^T(t)=\Big[x^T(t),f^T(x(t)),x^T(t-\tau(t)),x^T(t-\tau_1),x^T(t-\tau_2),$$
$$f^T(x(t-\tau(t))),\dot{x}^T(t-\tau(t)),\Big(\int_{t-\tau_2}^{t-\tau_1}x(s)ds\Big)^T,f_1^T,f_2^T,f_3^T\Big].$$

Notice the fact that $\Omega<0$ is equivalent to the LMI (7) by Lemma 2 (Schur complements). Thus, $V(t)<-\lambda\|x(t)\|^2$ for a sufficiently small $\lambda>0$. This ensures the global asymptotic stability of the considered system (4) with condition (2), which concludes the proof.

4 Numerical Examples

To illustrate the effectiveness of the theory developed in this paper, two numerical examples are proposed as follows:

Example 1. Consider a two-neuron neural network of neutral-type (4) with the following parameters:

$$A=\begin{bmatrix}2 & 0.1\\ 0.1 & 2\end{bmatrix},B=\begin{bmatrix}-0.7 & -0.2\\ -0.2 & -1.2\end{bmatrix},C=\begin{bmatrix}-1.2 & 1.1\\ 0.1 & -0.8\end{bmatrix},D=\begin{bmatrix}0.1 & 0\\ 0 & 0.1\end{bmatrix},$$
$$\alpha=0.1,\beta=0.08,\gamma=0.09.$$

By applying Theorem 1, one can see that the system (4) is asymptotically stable for $0<\tau(t)\le\tau_2$=0.7599.

Let τ_1=0, τ_2=0.7599, τ_d=0, by applying Theorem 1, there exists a feasible solution which guarantees the asymptotic stability of system (4):

$$P=10^4\times\begin{bmatrix}1.5773 & -0.0893\\ -0.0893 & 1.1773\end{bmatrix},Q_1=10^3\times\begin{bmatrix}2.4917 & 0.0483\\ 0.0483 & 2.8230\end{bmatrix},$$
$$Q_2=10^6\times\begin{bmatrix}2.8970 & -0.2478\\ -0.2478 & 1.9496\end{bmatrix},Q_3=10^3\times\begin{bmatrix}1.9629 & -0.1240\\ -0.1240 & 1.6420\end{bmatrix},$$
$$R_1=10^5\times\begin{bmatrix}4.6971 & -0.0975\\ -0.0975 & 4.7900\end{bmatrix},R_2=10^6\times\begin{bmatrix}3.1083 & -0.5813\\ -0.5813 & 0.7606\end{bmatrix},$$
$$R_3=10^3\times\begin{bmatrix}0.0635 & 0.1493\\ 0.1493 & 1.0844\end{bmatrix},Z_1=10^4\times\begin{bmatrix}2.2299 & -0.3520\\ -0.3520 & 1.2061\end{bmatrix},$$
$$Z_2=\begin{bmatrix}22.6386 & 42.7077\\ 42.7077 & 317.3317\end{bmatrix},K=10^3\times\mathrm{diag}\{8.3694 \quad 8.3694\},$$
$$\varepsilon_1=3.0726\text{e+}005,\varepsilon_2=3.8574\text{e+}005,\varepsilon_3=1.5106\text{e+}005.$$

Example 2. Consider the following three-neuron delayed neural network of neutral-type given in (4) with the following parameters:

$$A=\begin{bmatrix}1.2 & 0.1 & 0.2\\ 0.1 & 0.6 & 0.1\\ 0.2 & 0.1 & 0.7\end{bmatrix}, B=\begin{bmatrix}-0.2 & -0.1 & 0\\ 0.1 & -0.3 & -0.2\\ -0.2 & 0.1 & -0.12\end{bmatrix},$$

$$C=\begin{bmatrix}-0.1 & 0.2 & 0.2\\ -0.1 & -0.2 & 0\\ -0.2 & 0.1 & -0.2\end{bmatrix}, D=\begin{bmatrix}0.04 & -0.01 & -0.02\\ -0.01 & 0.09 & -0.05\\ -0.02 & -0.05 & 0.04\end{bmatrix},$$

$$\alpha=0.1,\beta=0.1,\gamma=0.2,\tau(t)=0.14+0.14\sin(t).$$

Let $\tau_1=0.14,\tau_2=0.28,\tau_d=0.14$. By applying Theorem 1, there exists a feasible solution which guarantees the asymptotic stability of system (4) as follows:

$$P=10^3\times\begin{bmatrix}5.5361 & 3.3970 & 4.3560\\ 3.3970 & 6.3503 & 3.5571\\ 4.3560 & 3.5571 & 4.4275\end{bmatrix}, Q_1=10^5\times\begin{bmatrix}6.1664 & 0.0550 & -0.2013\\ 0.0550 & 5.9567 & -0.0875\\ -0.2013 & -0.0875 & 6.2016\end{bmatrix},$$

$$Q_2=10^6\times\begin{bmatrix}3.3517 & -0.1462 & -1.4024\\ -0.1462 & 0.9348 & 0.0384\\ -1.4024 & 0.0384 & 2.0213\end{bmatrix}, Q_3=10^5\times\begin{bmatrix}5.1729 & -0.3673 & 1.2301\\ -0.3673 & 6.2866 & 0.4043\\ 1.2301 & 0.4043 & 4.9535\end{bmatrix},$$

$$R_1=10^5\times\begin{bmatrix}1.2630 & -0.3464 & 0.8691\\ -0.3464 & 2.6878 & 0.3975\\ 0.8691 & 0.3975 & 1.1128\end{bmatrix}, R_2=10^6\times\begin{bmatrix}0.9003 & 0.1181 & 0.1927\\ 0.1181 & 0.8527 & 0.1394\\ 0.1927 & 0.1394 & 1.6368\end{bmatrix},$$

$$R_3=10^4\times\begin{bmatrix}1.6774 & 0.4224 & 0.0368\\ 0.4224 & 1.4517 & 0.1573\\ 0.0368 & 0.1573 & 1.4035\end{bmatrix}, Z_1=10^6\times\begin{bmatrix}1.2030 & -0.2650 & 0.1819\\ -0.2650 & 1.0919 & 0.1541\\ 0.1819 & 0.1541 & 0.7136\end{bmatrix},$$

$$Z_2=10^5\times\begin{bmatrix}1.8099 & -0.4013 & 1.0528\\ -0.4013 & 2.6669 & 0.4050\\ 1.0528 & 0.4050 & 1.4461\end{bmatrix}, K=10^4\times\mathrm{diag}\{4.1118 \quad 4.1118 \quad 4.1118\},$$

$$\varepsilon_1=5.5317\text{e+}005,\varepsilon_2=5.8422\text{e+}005,\varepsilon_3=3.9861\text{e+}004.$$

5 Conclusion

A novel delay-dependent global asymptotic stability criterion for neural networks of neutral-type with interval time-varying delays and nonlinear perturbations has been provided. The new sufficient criterion has been presented in terms of LMI. The result is obtained based on the Lyapunov-Krasovskii method. The validity of the proposed approach has been demonstrated by two numerical examples.

References

1. Sun, C.Y., Zhang, K.J., Fei, S.M., Feng, C.: On exponential stability of delayed neural networks with a general class of activation functions. Physics Letters A 298, 122–132 (2002)
2. Zhang, Q., Ma, R.N., Chao, W., Jin, X.: On the global stability of delayed neural networks. IEEE Trans on Automatic Control 48(5), 794–7973 (2003)
3. Sun, C.Y., Feng, C.B.: Exponential periodicity and stability of delayed neural networks. Mathematics and Computers in Simulation 66(6), 469–478 (2004)
4. Liao, X.F., Li, C.D.: An LMI approach to asymptotical stability of multi-delayed neural networks. Physica D-Nonlinear Phenomena 200, 139–155 (2005)
5. Zhang, Q., Wei, X.P., Xu, J.: On global exponential stability of nonautonomous delayed neural networks. Chaos Soliton Fract. 26(3), 965–970 (2005)
6. Shen, T., Zhang, Y.: Improved global robust stability criteria for delayed neural networks. IEEE Trans. on Circuits and Systems Ii-Express Briefs 54(8), 715–719 (2007)
7. Wu, W., Cui, B.T.: Improved sufficient conditions for global asymptotic stability of delayed neural networks. IEEE Trans. on Circuits and Systems Ii-Express Briefs 54(7), 626–630 (2007)
8. Zhang, Q., Wei, X.P., Xu, J.: A new global stability result for delayed neural networks. Nonlinear Anal.-Real. 8(3), 1024–1028 (2007)
9. Wu, W., Cui, B.T.: Global robust exponential stability of delayed neural networks. Chaos Soliton Fract. 35(4), 747–754 (2008)
10. Sun, Y.H., Cao, J.D.: Novel criteria for mean square stability of stochastic delayed neural networks. Neural Network World 18(3), 245–254 (2008)
11. Yucel, E., Arik, S.: Novel results for global robust stability of delayed neural networks. Chaos Soliton Fract. 39(4), 1604–1614 (2009)
12. Hu, C., Jiang, H.J., Teng, Z.D.: Globally Exponential Stability for Delayed Neural Networks under Impulsive Control. Neural Process. Lett. 31(2), 105–127 (2010)
13. Cheng, C.J., Liao, T.L., Yan, J.J., Hwang, C.C.: Globally asymptotic stability of a class of neutral-type neural networks with delays. IEEE Trans. Syst Man Cybern. B Cybern. 36(5), 1191–1195 (2006)
14. He, H.L., Liao, X.X.: Stability analysis of neutral neural networks with time delay. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3971, pp. 147–152. Springer, Heidelberg (2006)
15. Park, J.H., Kwon, O.M.: Further results on state estimation for neural networks of neutral-type with time-varying delay. Appl. Math. Comput. 208(1), 69–75 (2009)
16. Liu, J., Zong, G.D.: New delay-dependent asymptotic stability conditions concerning BAM neural networks of neutral type. Neurocomputing 72, 2549–2555 (2009)
17. Park, J.H., Kwon, O.M.: Design of state estimator for neural networks of neutral-type. Appl. Math. Comput. 202(1), 360–369 (2008)
18. Wu, H.L., Zhong, Y., Mai, H.H.: On Delay-dependent Exponential Stability of Neutral-type Neural Networks with Interval Time-varying Delays. In: Proceedings of 2009 International Conference on Artificial Intelligence and Computational Intelligence, vol. Ii, pp. 563–569 (2009)
19. Park, J.H., Kwon, O.M.: Global stability for neural networks of neutral-type with interval time-varying delays. Chaos Soliton Fract. 41(3), 1174–1181 (2009)
20. Lien, C.H., Yu, K.W., Lin, Y.F., Chung, Y.J., Chung, L.: Exponential convergence rate estimation for uncertain delayed neural networks of neutral type. Chaos Soliton Fract. 40(5), 2491–2499 (2009)
21. Mahmoud, M.S., Ismail, A.: Improved results on robust exponential stability criteria for neutral-type delayed neural networks. Appl. Math. Comput. 217(7), 3011–3019 (2010)

Application of Generalized Chebyshev Neural Network in Air Quality Prediction

Fengjun Li

School of Mathematics and Computer Science,
Ningxia University, 750021 Yinchuan, P.R. China
muzi152148@163.com

Abstract. Air pollution time series is often characterized as chaotic in nature. The prediction using traditional statistical techniques and artificial neural network with back-propagation (BP) algorithm, which is most widely applied, do not give reliable prediction results. The new algorithm is therefore proposed to predict the chaotic time series based on the generalized Chebyshev neural network technique. In addition, the new algorithm has no problems such as local minima, slow convergence arising from the steepest descent-like algorithm. Finally, to illustrate the power of the Chebyshev Neural Network (CNN), a simulation example is presented to show good performance that extracts useful information from the weight functions for understanding relations inherent in the given patterns, and the trained CNN has good performance both on generalization and calculating precision.

Keywords: artificial neural network, Chebyshev polynomial, air quality prediction.

1 Introduction

More and more people today are paying attention to environmental quality, and numerous air environment studies have been carried out in various countries([1-3]). As a result, the air environmental quality should become an important issue of public interest. The prediction of air quality has become an important task due to its impact on human health, vegetation and environment. The common techniques applied in the literature to predict the air pollutant concentration are Box-Tenkis methods, transfer function models, regression techniques and ANNs ([3-5]). These techniques are either univariate or multivariate in nature. If the data on input variables are not available, univariate techniques are most preferable. Recently, the air pollution time series is characterized as chaotic in several research studies ([6,7]). The chaotic behavior is observed due to the interaction of air pollutants with many external variables such as meteorology. A common practice to predict the chaotic time series is by reconstructing the phase space of the system using delay space embedding given as

$$X_t = (x_t, x_{t-\tau}, x_{t-2\tau}, \cdots, x_{t-(n-1)\tau}), \tag{1}$$

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 472–479, 2011.
© Springer-Verlag Berlin Heidelberg 2011

where X_t denotes the reconstructed state space vector for the tth state, $x_t, t = 1, 2, \cdots, m$ is the observed time series, τ denotes the day time lag and n is the embedding dimension of the dynamical system.

Takes (1981) using the embedded theorem showed that a functional map exist that provides the next step predictions using the nonlinear predictive model

$$x(t+1) = F(x(t)). \tag{2}$$

Due to the nonlinear mapping capabilities and generalization abilities, multilayer perceptrons (MLPs) seem to be the appropriate choice for approximating this function ([8]). Several studies contribute towards this aspect showing the capabilities of MLPs in predicting the chaotic time series ([8,9]). This generally happens due to the fact that two chaotic time series can be similar point wise, but be produced by the different dynamical system or vice versa ([9]).

In this paper, a new technique is proposed to predict the chaotic time series using the artificial neural networks. This learning algorithm based on the generalized Chebyshev polynomial is introduced for training neural networks. As Chebyshev, the values of functions are found by the cubic spline interpolation for the given set of data point. The Chebyshev and the obtained value are the new training patterns of the networks. By using of the orthogonal property of Chebyshev polynomials, each weight function can be expressed as a generalized Chebyshev polynomial, therefore, the weight function is a optimal approximation polynomial in the least-squares sense.

2 Database

The proposed technique is used to model a well known chaotic time series based on Lorenz map. Lorenz attractor is given by the following equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = a(y-x)\frac{\mathrm{d}y}{\mathrm{d}t} = -xz + bx - y\frac{\mathrm{d}z}{\mathrm{d}t} = xy - cz, \tag{3}$$

where the parameters a, b and c are assigned the values as $a = 10, b = 28$ and $c = \frac{8}{3}$. This differential equation can be solved numerically using fourth-order Runge-Kutta integrator with a time step 0.01. The first 1000 samples of Lorenz map are plotted in Fig. 1.

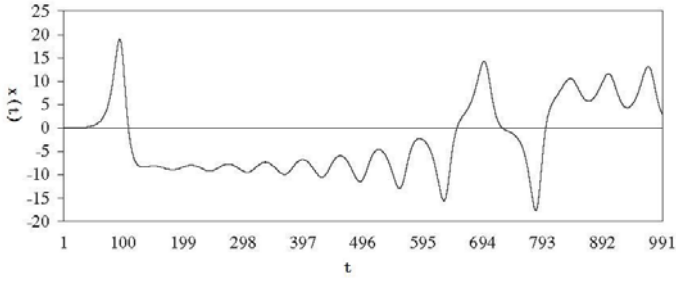


Fig. 1. NO_2: The first 1000 samples of Lorenz map

To examine the efficiency of proposed techniques, the real data of NO_2 concentration observed at a site in Ningdong Energy and Chemistry Industry Base are also used.

Yinchuan is the big city of western China. It is capital of Ningxia Hui Autonomous Region, and locates on upstream of Yellow River and middle of Ningxia plain. Region scope is in north latitude, east longitude. At present, Yinchuan urban district area is 1.5 million people, and 1,482 square kilometers, including three districts (Xingqing, Jinfeng and Xixia). It's also the most industrialized and populated district of Ningxia. Well known as "a land of rice and fish" in west-north of China, the region enjoys some of the favorable physical conditions, with a diversity of in natural resources and the suitability for growing various crops ([2]). The yearly average of NO_X has been substantially increased (about 3ug/m^3), since the beginning of monitoring in 2003; in the Yinchuan area. Suspended NO_X is mainly produced by chemical and physical mechanisms (about 66%), vehicular traffic (24%), moreover, a further significant part of NO_X is produced in the atmosphere because of residential heating (especially in winter)([3]).

Yinchuan is experiencing high levels of air pollution due to Chemistry Industry and coal fire. The distant has a arid climate with extreme weather conditions. Temperature inversion often occurs during winter months whereas during summer months, wind-blown dust covers the basin. this led to accumulation of air pollutants over the discrict. Many Chemistry Industries, power plants, coal mining also adds to increasing air pollution in the basin. The hourly data of nitrogen dioxide concentration observed at the Ningdong environmental center were obtained from continuous ambient air quality measurement analyzer. Fig. 2 shows the nitrogen dioxide time series observed during July and December 2008 in Ningdong Energy and Chemistry Industry Base. It can be seen that nitrogen dioxide concentration varies from 3 to 417 ug/m^3 with an average of $63ug/m^3$. High levels were observed between 6 a.m. and 10 p.m.

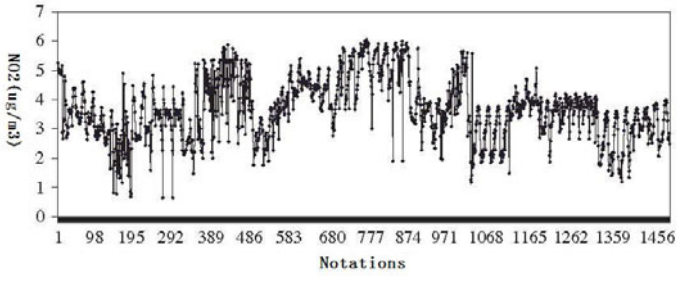


Fig. 2. The nitrogen dioxide time series observed at the Ningdong environmental center

3 Methodology

The introduced technique is based on an artificial neural network that with generalized Chebyshev polynomial as active function. In the new scheme of prediction, the training of weights and testing procedure are to be dealt separately. The algorithm for both the cases is given as follows.

3.1 Chebyshev Neural Network

CNN is a functional link network based on Chebyshev polynomials. One way to approximation a function by a polynomial is to use a truncated power series. The power series expansion represents the function with very small error near the point of expansion, but the error increases rapidly as we employ it at point farther away. The computational economy to be gained by chebyshev series increase when the power series is slowly convergent. Therefore, Chebyshev series are frequently used for approximations to functions and are much more efficient than other power series of the same degree. Among orthogonal polynomials, the Chebyshev polynomials occupy an important place, since, in the case of a broad class of functions, expansions in Chebyshev polynomials converge more rapidly than expansions in other set of polynomials. Hence, we consider the Chebyshev polynomials as basis function for the neural network.

The Chebyshev polynomials can be generated by the following recursive formula ([7]):

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x), \quad T_0(x) = 1. \tag{4}$$

For example, consider a two dimensional input patten $X = (x_1, x_2)^T$. An enhanced pattern obtained by using Chebyshev function is given as:

$$\varphi = (1, T_1(x_1)T_2(x_1), \cdots, T_1(x_2)T_2(x_2), \cdots)^T, \tag{5}$$

where $T_i(x_j)$ is a generalized Chebyshev polynomial, i the order of the polynomials chosen and $j = 1, 2$. The different choices of $T_1(x)$ are $x, 2x, 2x - 1$ and $2x + 1$. In this paper, $T_1(x)$ is chosen as x.

The output of the single layer neural network is given by:

$$\hat{y} = \hat{W}\varphi, \tag{6}$$

where $\hat{W}$ are the weights of the neural network given by $\hat{W} = (w_1, w_2, \cdots)^T$.

3.2 Learning Algorithm

The problem of identification consists in setting up a suitably parameterized identification model to optimize a performance function based on the error between the plant and identification model outputs. CNN, which is a single layered neural network is linear in the weights and nonlinear in the inputs is the identification model used in this paper. We shall use the recursive least squares method with forgetting factor as the learning algorithm for the purpose of on-line weight updation. The performance function to be minimized is given by:

$$E = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2. \tag{7}$$

The algorithm for the discrete time model is given by:

$$\begin{cases} \hat{W}(t) = \hat{W}(t-1) + n(t)e(t), \\ n(t) \quad = \frac{\lambda^{-1}P(t-1)\varphi(t)}{1+\lambda^{-1}\varphi^T(t)P(t-1)\varphi(t)}, \\ e(t) \quad = y(t) - \hat{y}(t), \\ P(t) \quad = \lambda^{-1}P(t-1) - \lambda^{-1}n(t)\varphi^T(t)P(t-1), \end{cases} \tag{8}$$

where λ is the forgetting factor and φ is the basis function formed by the functional expansion of the input and $P(0) = cI$, c is a positive constant, $||P(t)|| < M_0, M_0$ is a constant. All matrix and vectors are of compatible dimension for the purpose of computation.

The next pattern is then presented to the network and the above exercise is repeated. This exercise is continued for all the patterns to be presented to the network. For each pattern and each iteration, the error can be computed as the difference of network output and desired output, so the network can be trained on the error minimization criteria. The weights for each layer associated with each pattern should be stored in the memory. Once the network is trained for all the patterns, the next step is to examine its generalization capabilities.

4 Results and Discussion

The application of an ANN to the urban context in Yinchuan, particularly for the area of the Ningdong Energy and Chemistry Industry Base, near the Yellow river, is presented. The experimental data were obtained from the monitoring units of the Ningdong environmental center since 2008. The neural network was trained, tested and validated for many specific input configurations to forecast the concentrations of the single pollutant by varying the available information.

The computations were performed using MATLAB. For Lorenz map, the signal of first 1000 samples is obtained to consider as the training set, the next 250 samples are used as the testing set and subsequent 250 samples are considered as the prediction set. For nitrogen dioxide time series, the data available during November-December 2008 are divided into three parts. The data observed during November 2008 are considered for training, whereas next 15 days data are considered for testing and subsequent data are used for examining the prediction capabilities of the models. The next months data are divided equally in the testing and prediction sets. The testing and validation data should lie within the range of training data, which is evident from Fig. 2. The architecture of the CNN consist of two parts. namely numerical transformation part and learning part. Numerical transformation deals with the input to the hidden layer by approximate transformable method. The transformation is the functional expansion

of the input patten comprising of a finite set of Chebyshev polynomials. As a results the Chebyshev polynomial basis can be viewed as a new input vector. The CNN finally converged with the topology 5-8-1 and 6-14-1 for Lorenz and nitrogen dioxide time series, respectively.

For MLP with BP, the input to the network is the embedded space of time series with dimension and output is the next point in the time series. The network was trained with momentum constant and the learning rate was adoptive in this case. The number of hidden neurons was varied between 5 and 500. An error goal is specified as convergence criteria. With the variation in the number of hidden neurons and on convergence basis, the three-layer network with topology 5-10-1 and 6-15-1 is selected for Lorenz map and nitrogen dioxide time series prediction, respectively. The prediction results are plotted in Figs. 3 and 4.

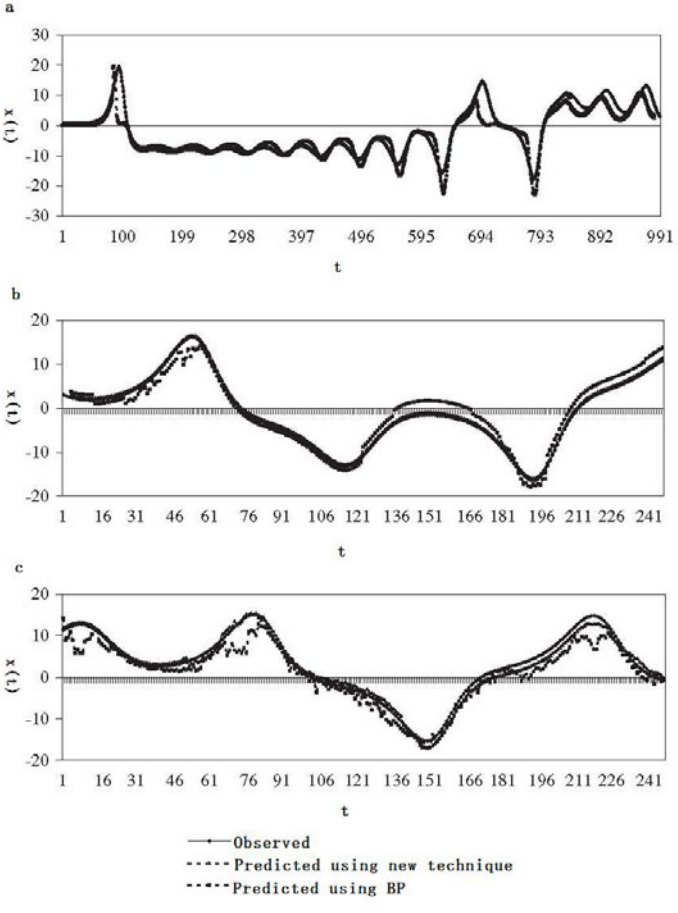


Fig. 3. The observed and predicted time series of Lorenz map: a, b, and c are the training set , the testing set and the prediction set, respectively

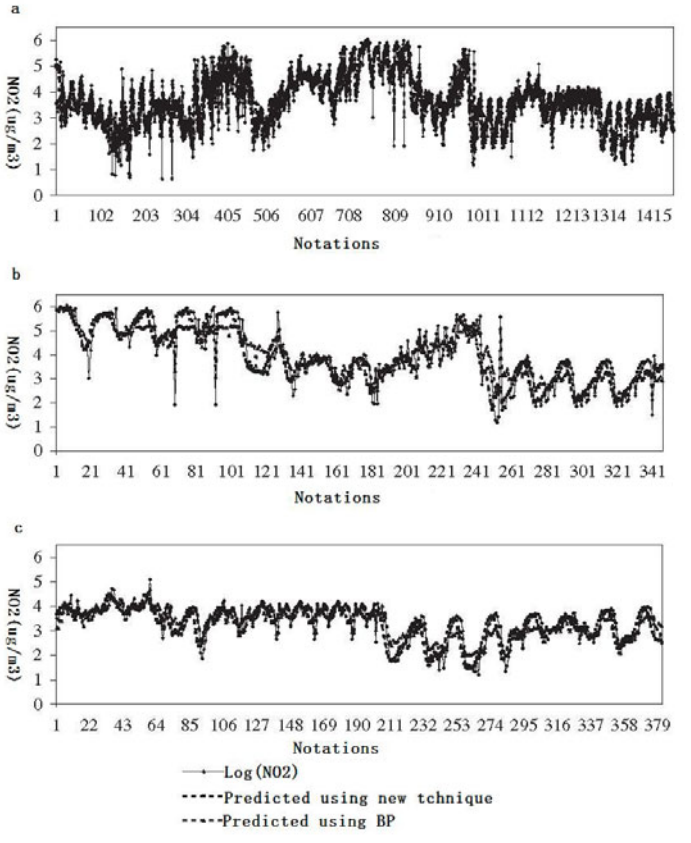


Fig. 4. The prediction of NO_2 time series using BP and new scheme: a, b, and c are the training set , the testing set and the prediction set, respectively

5 Conclusions

The CNN technique is proposed to predict the chaotic time series of air pollutant concentrations based on the neural network modeling technique. The applicability of the proposed scheme is demonstrated using a chaotic Lorenz map and nitrogen dioxide concentration time series. Comparing the results of MLP with BP and CNN techniquei, it is observed that the new scheme provides better predictions than the standard one with relatively less prediction error. Also, the invariant measures computed for the predicted time series using new scheme are very similar to the original one, whereas the predicted time series using MLP with BP do not posses the similar dynamic measures as the original one. Having the knowledge of internal dynamics enables the new approach to provide the prediction with better accuracy than the MLP with BP of prediction. Considering the chaotic nature of air pollutant concentration time series, the approach

seems to be very useful in predictions. Another advantage is the univariate nature of the approach, which takes into account the knowledge of embedding dimension. The scheme, however, requires more computational efforts in terms of computer memory and computational time. This result lead us to reduce the time requirements of the CNN in the future.

Acknowledgement

This work was supported by NSFC (61063020), Ningxia Ziran (NZ0907) and 2009 Ningxia Gaoxiao Foundations.

References

1. Duenas, C., Fernandez, M.C., Canete, S., Carretero, J., Liger, E.: Stochastic model to forecast ground-level ozone concentration at urban and rural area. Chemosphere 61, 137–1389 (2005)
2. Comrie, A.C.: Comparing neural networks and regression models for ozone forecasting. Journal of Air and Waste Management Association 47, 653–663 (1997)
3. Gardner, M.W., Dorling, S.R.: Neural network modeling and prediction of hourly NOx and NO2 concentrations in urban air in London. Atmospheric Environment 33, 709–719 (1999)
4. Sun, Y.Q., Hui, Q., Wu, X.H.: Hydrogeochemical characteristics of groundwater depression cones in Yinchuan City. Northwest China, Chinese Journal of Geochemistry 26, 350–355 (2007)
5. Sun, Y.C., Miao, O.L., Li, Y.C.: Prediction result analysis of air quality dynamic prediction system in Yinchuan city. Arid Meteorology 24, 89–94 (2006)
6. Han, Y.W., Gao, J.X., Li, H.: Ecology suitability analysis on the industry overall arrangement plan of Ningdong Energy Sources and Chemical Industry Base. Environmental Science and Management 32, 143–147 (2007)
7. Bishop, A.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
8. Wang, Y.M., Zhao, Y.F.: Issues of environmental protection and ecological construction of Ningdong Energy and Chemistry Industry Base. Ningxia Engineering Technology 7, 190–193 (2008) (in Chinese with English abstract)
9. Chu, G., Mi, W.B.: An analysis to the ecological carrying capacity of Yinchuan city. Urban Problems 10, 39–42 (2008) (in Chinese with English abstract)

Financial Time Series Forecast Using Neural Network Ensembles

Anupam Tarsauliya, Rahul Kala, Ritu Tiwari, and Anupam Shukla

Soft Computing & Expert System Laboratory, ABV - IIITM,
Gwalior-474010, India
{anupam8391,rkala001,drritutiwari,dranupamshukla}@gmail.com

Abstract. Financial time series has been standard complex problem in the field of forecasting due to its non-linearity and high volatility. Though various neural networks such as backpropagation, radial basis, recurrent and evolutionary etc. can be used for time series forecasting, each of them suffer from some flaws. Performances are more varied for different time series with loss of generalization. Each of the method poses some pros and cons for it. In this paper, we use ensembles of neural networks to get better performance for the financial time series forecasting. For neural network ensemble four different modules has been used and results of them are finally integrated using integrator to get the final output. Gating has been used as integration techniques for the ensembles modules. Empirical results obtained from ensemble approach confirm the outperformance of forecast results than single module results.

Keywords: Neural Network, Ensemble, BPA, RBF, RNN, Time Series.

1 Introduction

Financial forecasting is a standard benchmark example of time series analysis problem which is challenging due to high noise, volatility, and non-linearity [1]. These characteristics suggest that there is no complete information that could be obtained from the past behavior of financial markets to fully capture the dependency between the future price and that of the past. The domain contains some linear and nonlinear characteristics [2], and thus need to build a model is required which contains linear and nonlinear characteristics.

On the other hand, there is some risk to investment in the stock market due to its unpredictable behaviors. Thus, an intelligent prediction model for financial data would be of wider interest. ANNs are relatively recent method for business forecasting. The success of ANN applications can be qualified of their features and powerful pattern recognitions capability [3]. The use of ANN in this field has been growing due to their ability to model complex nonlinear systems on sample data. Back Propagation algorithm does the task of tuning of the ANN to enable it perform as desired [4]. Here the algorithm sets the various weights and biases of the ANN to their optimal values. The aim of training is to ensure that the network gives the closest desired outputs. The radial basis function network is another type of ANN that performs unsupervised or

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 480–488, 2011.
© Springer-Verlag Berlin Heidelberg 2011

supervised learning. These networks, which are known for their ability to model complex scenarios in a very simple architecture, are used for classification, series prediction, control applications, and so forth. The radial basis function network uses radial basis functions as activation functions, which allows it to easily solve the problem or perform a mapping of the inputs to outputs [5].

Recurrent neural networks are a special type of ANN in which the output of one or more neurons is fed back into the network as inputs, forming the input for the next iteration. These networks offer a good means of machine learning in which past outputs may potentially affect the next outputs [6].The conventional training algorithms of the neural networks are prone to get struck at local minima. This is because of the gradient approaches, or similar approaches of tuning the parameters that they employ. Genetic Algorithms [7] use a variety of individuals, each of which presents an ANN. Each of the ANN or individual represents a point in the error space with some degree of performance or fitness value. These algorithms use the relative fitness or performance to generate or move the various individuals in this error space to finally enable convergence at the optimal value.

2 Algorithms and Methods

2.1 Backpropagation Algorithm

General BPA Neural Network architecture shown in Fig. 1 includes input layer, hidden layer and output layer. Each neuron in input layers are interconnected with neurons in hidden layers with appropriate weights assigned to them. Similarly each neuron of hidden layer in interconnected with output layer neuron with weights assigned to the connection. On providing learning data to the network, the learning values are passed through input to hidden and finally to output layer where response for input data is obtained. For optimizing the error obtained, the error values are back propagated to make changes in weights of input to hidden layer and hidden to output layer. With error back propagation input response are made converged to desired response [4] [8]. A general structure of BPA neural network has been shown below.

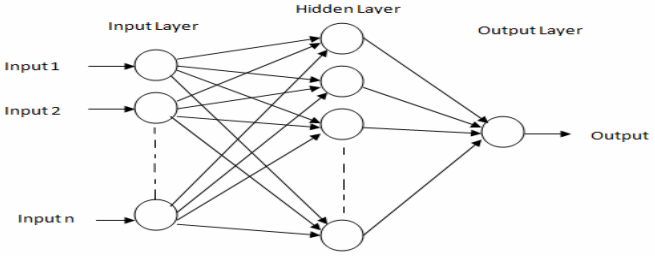


Fig. 1. General architecture of a Backpropagation Neural Network

2.2 Radial Basis Neural Network

Radial Basis Function Networks [5] has a simple 3 layer ANN architecture. The first layer is the input layer where the inputs are applied. The second layer is the hidden

layer. The last layer is the output layer where the system delivers its output. These networks however differ in the manner in which they process information for the generation of the outputs from the inputs.

Consider the input space of the problem being considered. Each neuron of the hidden layer in the RBN corresponds to a location in the input space. The various neurons are spread all round the input space. The input itself is a point in this input space consisting of some value of different attributes. At the time of processing of the inputs, each of these neurons calculates its distance from the input. The outputs of the various neurons of the hidden layer serve as the inputs of the output layer. This layer is a linear layer and simply computes the weighted sum of the various hidden layer neurons. Each connection of this layer corresponds to the weight that is multiplied by the associated hidden neuron. In this way the system generates the final output.

2.3 Recurrent Neural Network

Recurrent Neural Networks architecture shown in Fig. 2 [6] are cyclic in nature unlike the other models that were arranged in a manner that cycles can never be formed. The conventional ANNs have a big limitation of their static nature. The information flow is only forward where the predeceasing layers process data and forward it to the next layers. These networks allow backward connections where every neuron gets the feedback from the forward layers as well as itself [11]. This allows the ANN to again process data and again transmit the output for further processing by the other layers in backward and forward direction to which it is connected. In this manner there is a lot of dynamism which drives these networks. Further the algorithm operates in time-stamps or iterations where a unit processing is performed by each of the neurons in a single iteration. The output of the system continuously changes with time as the layers undergo changes driven by the feedback connections.

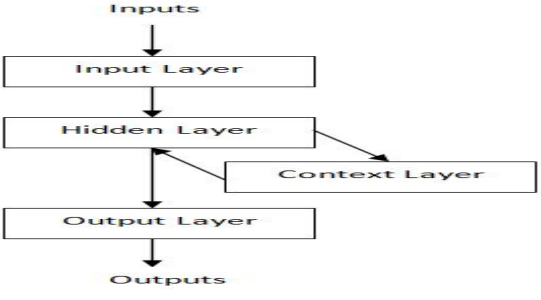


Fig. 2. General architecture of a Recurrent Neural Network

2.4 Evolutionary BPA Neural Network

A genetic algorithm performs a parallel stochastic search. It is parallel in the sense that many solutions in the population are considered simultaneously and the fittest solutions are chosen for reproduction. It is stochastic in the sense that the solutions are randomly selected for refinement and the likelihood of a solution being selected is

enhanced by the quality of the solution or its fitness, and the search direction is also chosen randomly. Genetic Algorithm evolves ANNs by fixing the values and the weights and biases of the various nodes i.e. the GA optimizes the network parameters for better performance [13].

Steps followed for evolution of ANN [12][13] are problem encoding, creation of random initial state, fitness evaluation, and genetic operator including selection, crossover, mutation and elite, generate next generation, testing and verification.

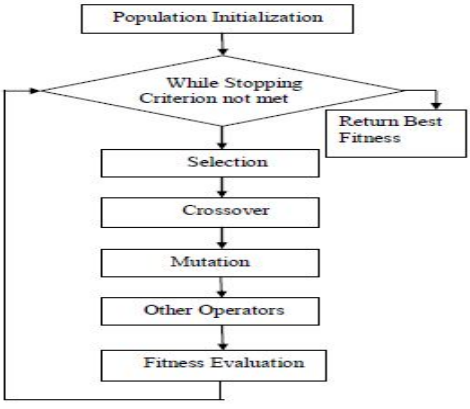


Fig. 3. Flow Diagram for working of Genetic Algorithm

2.5 Ensembles

Ensembles make use of multiple ANNs to solve the same problem. Each of the ANN is given the complete part of the problem input. Each ANN solves the problem using its own approach. All these ANNs or modules return a solution to the integrator which then computes the final output [14]. The input is given by the system to each of the modules. Each module is similar in regard to the inputs and the outputs. Each module represents an independent ANN of its own which is trained separately using the same training data. The training may hence be carried out in parallel among the various ANNs or modules. In this manner all the modules or ANN solve the same problem and make their distinct contributions. All the outputs are communicated to a central integrator that carries the next task of computing the final output of the system based on the system module outputs. Justification to ensembles revolve around the fact that they are more resistant to accidental under-training of a single ANN, or it being accidentally struck at a local minima, since the effect may be averaged by others.

In gating as integration technique [15], we try to combine the various outputs of the modules and generate the final output of the system by a simple function. The easiest implementation of this is to make the integrating function take a weighted or simple mean of the various outputs of the modules. This gives the final system output of the system.

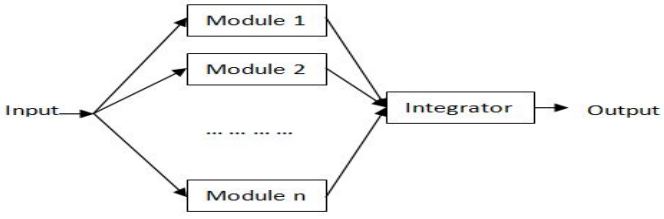


Fig. 4. General Architecture of a Neural Network Ensemble

3 Experiment and Results

3.1 Research Data

We have used two different data sets for our research. The data (un-normalized) have been collected from Prof. Rob J Hyndman's website http://robjhyndman.com/TSDL/ . Data sets analyzed are as: Daily closing price of IBM stock, Jan. 01 1980 - Oct. 08 1992 [10], Daily S & P 500 index of stocks, Jan. 01 1980 - Oct. 08 1992 [10]. The first few data indexes of series are used for the research. For training, 80% of the data of the series has been used and remaining 20% is used for testing.

Table 1. Time Series Data Sets Description

Time Series	Standard Deviation	Mean	Count
Daily IBM	5.736916	60.89908	500
Daily S&P	10.1308	123.3728	500

3.2 Methodology

Time series dataset is divided into training and testing dataset. Dataset division carried out is based on the random probability followed which bifurcates the dataset into training and testing dataset. Training dataset which is 80% of the original dataset is used for defining and training of the neural network. Testing dataset which is remaining 20% of the original dataset is used for performance measure. Performance measurement carried out in the experiment is root mean square error.

After dataset division, normalization followed by logarithmic scale conversion is carried out to draw better timely relation between index values. Thus processed training dataset is used for defining and supervised learning of the neural network. For whole experiment, total number of input neurons and total number of output neuron are kept 10 and 01 respectively. Neural Network training followed by testing is carried out one by one on all four modules to record the individual performance for the individuals. Corresponding graphs between actual and predicted values are plotted to analyze the performance graphically.

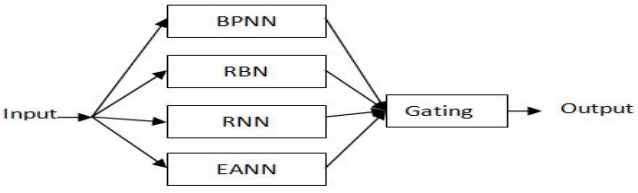


Fig. 5. Used Architecture of Neural Network Ensemble

For neural network ensemble used architecture as shown in figure 5, backpropagation, radial basis, recurrent and evolutionary neural networks are considered as individual modules of ensemble. Each modules gives output for given input to the ensemble system. Output of these modules as discussed is integrated by gating technique. In gating, weighted mean is followed for outcomes as final output result. Thus found outcome of the neural network ensemble is compared with the individual performance of standalone modules. Graph for actual and predicted values is also plotted for neural network ensemble for carrying out the graphical analysis of individual and ensemble performances.

3.3 Empirical Results

Table 2 shown below represents the mean root mean square error for both of the time series Daily IBM and Daily S&P.

Table 2. Results Obtained for Time Series

Methodology	Daily IBM (Mean RMSE)	Daily S&P (Mean RMSE)
BPA	1.9823	2.8913
RBN	1.2261	1.5217
RNN	0.9918	1.1371
EANN-BPA	1.4725	2.1237
Ensemble	0.9743	1.1261

3.4 Graphical Analysis

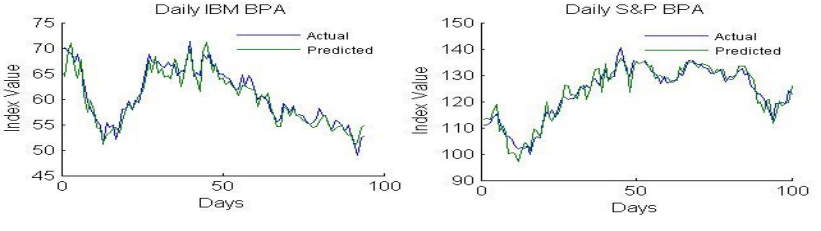


Fig. 6. Graphs for Actual and Predicted Values for Daily IBM and Daily S&P using traditional Backpropagation Algorithm with mean RMSE = 1.9823 and 2.8913 respectively

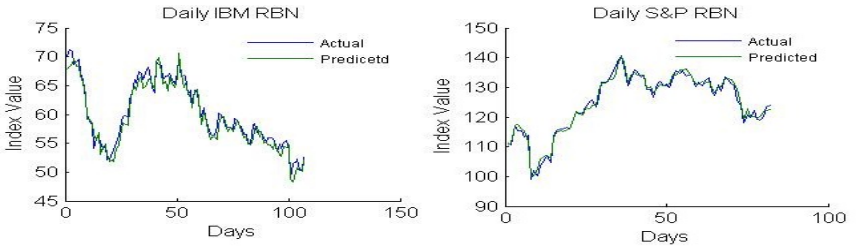


Fig. 7. Graphs for Actual and Predicted Values for Daily IBM and Daily S&P using Radial Basis Network with mean RMSE = 1.2261 and 1.5217 respectively

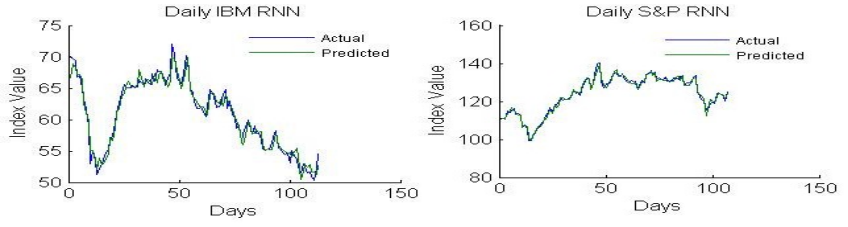


Fig. 8. Graphs for Actual and Predicted Values for Daily IBM and Daily S&P using Recurrent Neural Network with mean RMSE = 0.9918 and 1.1371 respectively

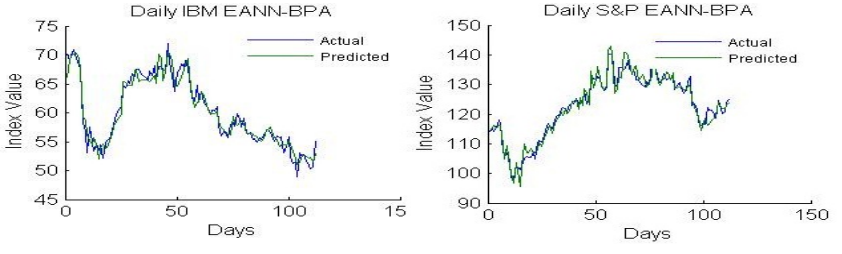


Fig. 9. Graphs for Actual and Predicted Values for Daily IBM and Daily S&P using Evolutionary BPA Neural Network with mean RMSE = 1.4725 and 2.1237 respectively

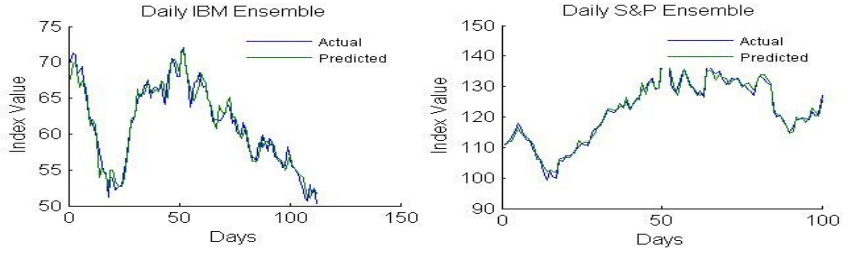


Fig. 10. Graphs for Actual and Predicted Values for Daily IBM and Daily S&P using Neural Network Ensemble with mean RMSE = 0.9743 and 1.1261 respectively

4 Conclusions

Neural network ensemble has been used in order to improve the financial forecasting performance. It is based on the taking different learning models as individual modules, where each module accounts for the performance of the final result of the system. Output values from the individual modules are integrated using gating as integration technique in order to have the final output results. Justification to ensembles revolve around the fact that they are more resistant to accidental under-training of a single ANN, or it being accidentally struck at a local minima, since the effect may be averaged by others. As a performance measurement root mean square error is used. Neural network ensembles are computationally efficient and it shows a very good behavior to estimation values. Accuracy results and comparison graph of actual and predicted index values shows the better performance of neural network ensemble over performances of individual module used. Order of performance on the basis of results can be adjudged as Ensembles > RNN > RBN > EANN-BPA > BPA.

References

1. Wang, D., Li, Y.: A novel nonlinear RBF neural network ensemble model for financial time series forecasting. In: 2010 Third International Workshop on Advanced Computational Intelligence (IWACI), August 25-27, pp. 86–90 (2010)
2. Zhou, H.-r., Wei, Y.-h.: Stocks market modeling and forecasting based on HGA and wavelet neural networks. In: 2010 Sixth International Conference on Natural Computation (ICNC), August 10-12, vol. 2, pp. 620–625 (2010)
3. Ning Y.-c., Zheng X.-x., Zhao, J., Jiang, G.-j.: Forecasting the natural forest stand age based on artificial neural network model. In: 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering (CCTAE), June 12-13, vol. 3, pp. 536–539 (2010)
4. Zhai, F., Wen, Q., Yang, Z., Song, Y.: Hybrid forecasting model research on stock data mining. In: 4th International Conference on New Trends in Information Science and Service Science (NISS), May 11-13, pp. 630–633 (2010)
5. Karimi, B., Menhaj, M.B., Saboori, I.: Robust Adaptive Control of Nonaffine Nonlinear Systems Using Radial Basis Function Neural Networks. In: 32nd Annual Conference on IEEE Industrial Electronics, IECON 2006, November 6-10, pp. 495–500 (2006)
6. Gupta, L., McAvoy, M., Phegley, J.: Classification of temporal sequences via prediction using the simple recurrent neural network. Pattern Recognition 33(10), 1759–1770 (2000) ISSN 0031-3203
7. Shopova, E.G., Vaklieva-Bancheva, N.G.: BASIC–A genetic algorithm for engineering problems solution. Computers & Chemical Engineering 30(8), 1293–1309 (2006) ISSN 0098-1354
8. Lee, T.-L.: Back-propagation neural network for the prediction of the short-term storm surge in Taichung harbor. Engineering Applications of Artificial Intelligence 21(1), 63–72 (2008) ISSN 0952-1976
9. Casasent, D., Chen, X.-w.: Radial basis function neural networks for nonlinear Fisher discrimination and Neyman-Pearson classification, Neural Networks. In: Advances in Neural Networks Research: IJCNN 2003, June-July 2003, vol. 16(5-6), pp. 529–535 (2003) ISSN 0893-6080

10. Hipel and McLeod Time Series Modelling of Water Resources and Environmental Systems. Elsevier, Amsterdam (1994)
11. Assaad, M., Bone, R., Cardot, H.: A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Information Fusion, Special Issue on Applications of Ensemble Methods 9(1), 41–55 (2008) ISSN 1566-2535
12. Guo, Y., Kang, L., Liu, F., Sun, H., Mei, L.: Evolutionary Neural Networks Applied to Land-cover Classification in Zhaoyuan. In: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2007, March 1-April 5, pp. 499–503 (2007)
13. Sexton, R.S., Gupta, J.N.D.: Comparative evaluation of genetic algorithm and backpropagation for training neural networks. Information Sciences 129(1-4), 45–59 (2000) ISSN 0020-0255
14. Tsai, C.-F., Wu, J.-W.: Using neural network ensembles for bankruptcy prediction and credit scoring. Expert Systems with Applications 34(4), 2639–2649 (2008) ISSN 0957-4174
15. Oza, N.C., Tumer, K.: Classifier ensembles: Select real-world applications. Information Fusion, Special Issue on Applications of Ensemble Methods 9(1), 4–20 (2008) ISSN 1566-2535

Selection of Software Reliability Model Based on BP Neural Network

Yingbo Wu[1] and Xu Wang[2]

[1] School of Software Engineering
[2] School of Mechanical Engineering,
Chongqing University, Chongqing 400044, China
wyb@cqu.edu.cn, wx921@163.com

Abstract. Software reliability models are used for the estimation and prediction of software reliability. In a situation where reliability data is lacking and numerous models are available, the key to quantitative analysis of software reliability lies in the selection of an optimal model. This paper describes a model selection method which involves an encoding scheme with multiple evaluation metrics and uses back-propagation (BP) neural network to perform clustering algorithm. Finally, by utilizing 20 sets of failure data that are collected in actual software development projects, a simulation experiment is made. The result shows the method is both correct and feasible.

Keywords: software reliability model, BP neural network, model selection.

1 Introduction

Software quality refers to the totality of features and characteristics of a software product that bear on the ability to satisfy specific needs. Among them, the software reliability, as the inherent feature of software quality, has become an important parameter in software quality assessment because of its wide coverage, ready quantization and close association with other software quality features. Software reliability control and assessment must be implemented with a software reliability model. Since the initial model was introduced in 1970s, hundreds of models have been proposed. However, up until now, no general model has been confirmed that can be applied to all software products.

The inconsistent results yielded by applying different software reliability models derive from the different assumptions underlying every model. Because each reliability model has its own specific theoretical basis, thus different models deduce and calculate reliability in different ways, which in turn produces different reliability assessment results. At the moment, it is difficult to put forward a universal model. Besides, the universal model only adds more complexity to inconsistency in model application. Therefore, the most feasible way to solve the inconsistency issue is to select optimal model and combine different models. As far as software technicians are concerned, in a situation where reliability data is lacking and numerous models are available, how to select appropriate model to better ensure software reliability has become a heated topic.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 489–496, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Selecting a reliability model is a complex decision which should be based on multiple criteria [1]. [2] introduces the concept of expert system on reliability, and defined model selection as an optimization issue based on preliminary result from user's software failure data and bound by preset optimization principles. Currently, the selection of software reliability model is primarily based on observer's personal experience. By observing actual testing performance, modifications are made in software reliability growth model. [3] describes the unascertained-based software reliability model selection, and at this basis, an automated tool for selecting reliability model is proposed. [4] presents a meta-learning approach and describes experimental results from the use of a neural network meta classifier for selection among different kind of reliability models. [5] has offered a new insight into reliability model selection: it is generally considered appropriate to select the same model for similar failure data sets. To be specific, different software failure data are sent to a classifier where cluster takes place, and software thus classified into the same category adopt the same model. Inspired by clustering idea, a means to select reliability model has been proposed which takes advantage of classifying ability of BP (back propagation) neural network.

2 BP Neural Network Model

2.1 Mechanism of BP Neural Network

Artificial neural network is a technological system which simulates the structure and functions of neural networks in human brain by using engineering technology. Numerous nodes are abstracted from human brain, which are termed as processing unit. These processing units connect with each other to form a neural network.

Because of the particularity of neural network technology, it plays significant roles in the field of software reliability, and has broad prospects [6]. Mechanism of the neural network is illustrated in Figure 1.

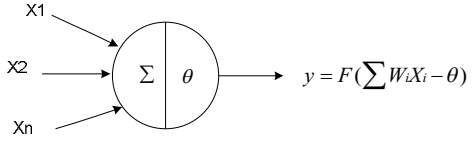


Fig. 1. Mechanism of the Neural Network

In Figure 1, X_1, X_2, X_n is the input neuron, which refers to the information from axons of n neurons at the early level. A is the threshold of i neurons; W1, W2, Wn i neurons express weight coefficient of neuron i to X1, X2, Xn, namely, the efficiency of synaptic transmission; Y1 is the output of i neurons; f [•] express an activation function which determines the manner of the output when i neurons reach its threshold by the concerted stimulation from Inputs X1, X2, and Xn.

BP neural network adopted in this paper is a kind of forward neural network without feedback. It can establish the global nonlinear mapping between input and output variables through learning from certain samples. It is by far the most widely used

neural network, which is so named because it adopts back propagation proposed in 1986 by Rumelhart et al. BP algorithm is a learning algorithm for feed-forward multi-layer network which intends to solve weight coefficient optimization of multilayer forward neural network. Its network architecture is shown in Figure 2.

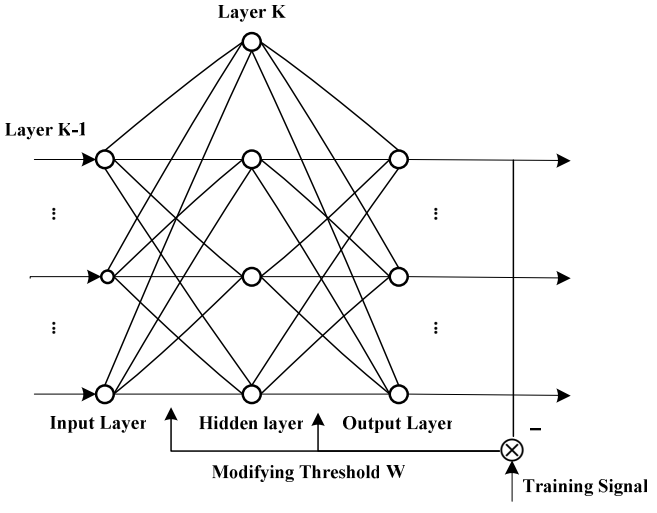


Fig. 2. Network Structure of BP

It contains the input layer, the output layer and the intermediate layer between the input and output layers. The intermediate layer is single-or multi-layered, which is also referred to as hidden layer because of its non direct contact with the outside world. By the same token, neurons in the hidden layer are known as hidden units. Although the hidden layer has no contact with the outside world, its state of affair will affect the relationship between input and output. In other words, weight coefficient changes in the hidden layer will result in changes in the overall performance of the multilayer neural network. Suppose there exists an M-layered neural network with Sample X at the input layer; the sum total of i neurons input at the k layer is expressed as U_i^k , and the output is expressed as X_i^k ; weight coefficient of j neurons at Layer k-1 to i neurons at Layer k is referred to as W_{ij}, and the activation function of each neuron is f, then the relationship between variables can be expressed in a mathematical formula as (1):

$$X_i^k = f(U_i^k), U_i^k = \sum_j Wij X_j^{k-1} \tag{1}$$

Back-propagation algorithm involves two processes, namely, forward propagation and back propagation. These two processes are summarized as follows:

1) Forward propagation. Input samples are processed at hidden units at all hidden layers. After that, they are sent to the output layer. In the course of hierarchical

processing, neurons at each layer only influence neurons at the next level. At the output layer, comparison is made between the actual output and the estimated output. If the actual output is inconsistent with the estimated output, then back propagation comes into play.

2) Back-propagation. In back propagation, error signal is transmitted in a way that is exactly opposite to the forward propagation. Adjustments are made to the weight coefficient of every neuron at every hidden layer so that error signals are reduced to the minimum.

In essence, BP algorithm is to obtain the minimum value of error function. This algorithm adopts the steepest descent method in nonlinear programming and adjusts weight coefficient with gradient descent approach of error function.

2.2 Learning Steps in BP Algorithm

When back-propagation algorithm is applied to multiplayer feed forward network, the following steps can be taken to obtain the network weight coefficient by recursion. Note that for each layer with n neurons, and for Neuron i at Layer K, then n weight coefficients $W_{i1},W_{i2},\ldots,W_{in}$ are available, a further $W_{i,n+1}$ expresses threshold θ_i ; besides, when sample X is input, then $x=(X_1,X_2,\ldots,X_n,1)$. The algorithm takes the following steps:

1) Set initial weight coefficient of Wij. Set a smaller non-zero random number for Wij at each layer, where $W_{i,n+1}=-\theta$.
2) Input a sample $x=(X_1,X_2,\ldots,X_n,1)$ and the corresponding estimated output $Y=(Y_1,Y_2,\ldots,Y_n)$.
3) Calculate the output at each layer. For the output of neuron i at Layer K, then

$$U_k^i = \sum_{i=1}^{n} W_{ij} X_j^{k-1}, X_{n+1}^{k-1} = 1, W_{i,n+1} = -\theta, X_i^k = f(U_i^k) \tag{2}$$

4) Calculate the learning errors at all layers d_i^k . For the output layer m then

$$d_i^m = X_i^m (1 - X_i^m)(X_i^m - Y_i) \tag{3}$$

while for the other layers, then

$$d_i^k = X_i^k (1 - X_i^k) \sum_i W_{ij} d_j^{k+1} \tag{4}$$

5) Adjust weight coefficient W_{ij} and the threshold θ. Then we get

$$W_{ij}(t+1) = W_{ij}(t) - \eta \cdot d_j^k \cdot X \tag{5}$$

Where

$$\Delta W_{ij}(t) = -\eta \cdot d_i^k \cdot X_j^{k-1} + a\Delta W_{ij}(t-1) = W_{ij}(t) - W_{ij}(t-1) \quad (6)$$

6) When weight coefficient at each layer is obtained, judgment can be made on whether requirements are met based on given quality index. If met, the algorithm comes to an end; if not, return to step 3).

3 The Execution Process of Model Selection Based on BP Network

It is inappropriate to directly use failure data as BP network input data, before failure data is encoded [7]. The encoded data are then clustered through a BP network which is stabilized by supervised learning. All these done, optimal model selection can be achieved.

3.1 Software Reliability Model

The three models proposed in [8] serve as the candidate model, referred to as JM, GO, and LV respectively. These three models are widely used in reliability analysis. In practice, candidate models can be any proposed reliability model. For sake of convenience, this paper employs the above-mentioned three models with data in [7] as comparison data. Among the three models, JM is a mathematical model of Markov process, which is the most representative among the first software reliability models; GO is the best-known NHPP process model; and LV is a typical Baysian model. These three models include both selective process model and non-selective process model. Thus, it can be drawn that they generally represent the features of most software reliability models and comparisons can be made between them. The encoded software failure data will be divided into three categories, with each category corresponding to these three models.

3.2 Encoding

In order to compare application ability among software reliability models, the researchers have put forward many evaluation metrics. This paper uses 5 mature ones, namely: GOF(good-of-fit), PL(prequential likelihood), MB(model bias), MBT(model bias trend), and MN(model noise). The definition and calculation method of these evaluation metrics can be found in [8]. This paper will not discuss them for lack of space. The three models are encoded with 1 for JM, 2 for GO, and 3 for LV.

Coding steps as following:

1) For given software failure data, GOF values of the three models are calculated. Based on GOF values, the models are arranged by voting. The model with the best GOF scores 3 points, the second best 2 points, and the worst 1 point. The model with the highest score is coded as x_1;
2) x2, x3, x4, x5 are obtained by calculating the values of the remaining evaluation metrics in the same way as step 1);
3) The vector (x_1, x_2, x_3, x_4, x_5) is the code that corresponds to specific software failure data.

Table 1 illustrates Data Set J2 of Project JPL in [9], with numbers in brackets indicating the score, the encoding results are expressed as {1,3,3,3,2}.

Table 1. Encoding of Data Set J2

Evaluation Principles	JM Model	GO Model	LV Model	Scoring No.
Good-of-Fit	0.162(3)	0.182(1)	0.17(2)	1
Accuracy	1074(2)	1075(1)	1051(3)	3
Model Bias	0.3378(2)	0.3378(2)	0.2592(3)	3
Model Bias Trend	0.4952(2)	0.4954(1)	0.1082(3)	3
Model Noise	32607(2)	2539(3)	2333(1)	2

3.3 The Model Selection Process

Firstly, baseline data encoding offers learning module for BP network. This paper employs three models of JM, GO, and LV, thus the baseline data are encoded as {1,1,1,1,1}, {2,2,2,2,2} and {3,3,3,3,3} to represent respectively appropriate models of JM (1), GO (2), and LV (3). Obviously, if the first encoded vector is input, the estimated result by BP network computing should be 1. Similarly, the estimated output of second input vector should be 2. And the estimated output of the third vector should be 3. Based on learning algorithm stated in Section 2.2, calculations continue until the probability of getting correct (estimated) output stabilizes. With BP network being stable and weight coefficient value W_i being optimal, at this point, learning ends and actual calculations come into play. Next, encode the actual failure data. Then, the encoded data are input into the stabilized BP network to receive clustering algorithm. Finally, by referring to the above-mentioned model numbers of the three models, judgment can be made on which software reliability model is most suitable for the kind of software with given failure data.

4 Simulation Analysis

We make the simulation by using 20 sets of failure data that are collected in actual software development projects. These 20 data sets originate from Data Set DACS published 1979 by Musa [7]. DACS are reasonable with good performance. The 20 data sets are encoded as Data1, Data2, ···, Data20. Among them, Data1, Data2 and Data3 are defined as baseline data, coded as {1,1,1,1,1}、{2,2,2,2,2} and {3,3,3,3,3}. Table 2 lists the results obtained by this approach and the cumulative ranking results commonly used in engineering. By comparison, it can be found that the results in Data7 and Data14 differ with each other. Therefore, the approach proposed in this paper has certain reliability. The use of Matlab neural network toolbox makes possible complex calculations in BP network, and ensures accuracy in calculation.

Compared with Kohonen network-based approach in [7], BP network is advantageous in that it is a supervised learning approach, where empirical data are adopted as sample to guide network learning. It is more stable than unsupervised autonomous learning. Besides, the error ratio of this approach relative to cumulative ranking

approach is 10%; while those of Gaussian Mixture Model as in [1] and comparative error coefficient as in [10] are 15% and 23% respectively. This also confirms the validity and accuracy of this approach.

Table 2. Classification of Simulation Data

Data No.	Coding Result	Accumulation Ordering	Computing Result of Neural Network
1	11111	JM(1)	JM(1)
2	22222	GO(2)	GO(2)
3	33333	LV(3)	LV(3)
4	33312	LV(3)	LV(3)
5	21131	JM(1)	JM(1)
6	21131	JM(1)	JM(1)
7	21121	JM(1)	GO(2)
8	23111	JM(1)	JM(1)
9	22221	GO(2)	GO(2)
10	33131	LV(3)	LV(3)
11	13233	GO(3)	GO(3)
12	21211	JM(1)	JM(1)
13	33322	LV(3)	LV(3)
14	22311	GO(2)	JM(1)
15	12311	JM(1)	JM(1)
16	12313	JM(1)	JM(1)
17	21213	GO(2)	GO(2)
18	23333	LV(3)	LV(3)
19	11123	JM(1)	JM(1)
20	13332	LV(3)	LV(3)

5 Conclusions

Extensive researches at home and abroad have indicated that several evaluation metrics should be taken into account in determining model application. This paper involves an encoding scheme with multiple evaluation metrics and uses neural network to perform clustering algorithm, thus offering new insights into model selection. This new approach is characterized by high precision, simple structure, and fast calculating speed. However, this approach is not without flaws. First of all, 17 failure data sets are used in this paper and baseline data have involved in supervised learning for nearly 80 times. Although the accuracy has somewhat been verified in simulation experiment, more failure data should be used and more learning be carried out to further enhance the accuracy of the network. Next, BP network learning is an important step, where learning sample is essential to model stability and accuracy. The essence of this approach is to automatically optimize standard weight coefficients through sample learning by BP network. Therefore, in order to strengthen the accuracy and stability of the network clustering, further study is needed on how to collect and select empirical data to supervise network learning. Finally, as model selection cannot be separated from model evaluation metrics, research on and advancement of more effective model evaluation metrics will further improve model selection approaches.

References

1. Wang, H., Liu, C., Jin, M.: Software Reliability Growth Model Selection and Combination: A New Approach. Computer Engineering and Applications 21(4), 21–24 (2002)
2. Str Ingfellow, C., Andrew, S.A.: An Empirical Method for Selecting Software Reliability Growth Models. Journal of Empirical Software Engineering 7(4), 319–343 (2002)
3. Zhang, Y., Liu, Y.: Research on Software Reliability Model Selection Based on Unascertained Set. Systems Engineering Theory and Practice 26(8), 91–94 (2006)
4. Caiuta, R., Pozo, A., Emmendorfer, L., Vergilio, S.R.: Selecting software reliability models with a neural network meta classifier. In: IEEE International Joint Conference on Neural Networks(IJCNN 2008), pp. 3747–3754. IEEE Press, New York (2008)
5. Xie, M., Hong, G.Y., Wohlin: Software reliability prediction incorporating information from a similar project. Journal of System and Software 3(5), 43–48 (1999)
6. Karunanithi, N., Whitley, D., Malaiya, Y.K.: Using Neural Networks in Reliability Prediction. IEEE Trans. on Software Engineering 9(4), 53–59 (1992)
7. Wu, Q., Hou, C., Yuan, J.: Selection of software reliability model based on Kohonen network. Computer Applications 25(10), 2331–2333 (2005)
8. Michaelrl: Handbook of Software Reliability Engineering. IEEE Computer Society Press, New York (1996)
9. Mussa J.D.: Software Reliability Data. DACS, RADC, `http://www.dacs.dtic.Mil/databases/sled/swrel.shtml`
10. Song, X.: Software Reliability & Maintainability Evaluation Tool. Electronic Product Reliability and Environmental Testing 24(3), 16–21 (1999)

Atavistic Strategy for Genetic Algorithm

Dongmei Lin[1], Xiaodong Li[2], and Dong Wang[3]

[1] Center of Information and Education Technology,
Foshan University, Foshan 528000, China
[2] School of Computer Science and Information Technology,
RMIT University, Melbourne, Australia
[3] Department of Computer Science and Technology,
Foshan University, Foshan 528000, China
{dmlin,wdong}@fosu.edu.cn, xiaodong.li@rmit.edu.au

Abstract. Atavistic evolutionary strategy for genetic algorithm is put forward according to the atavistic phenomena existing in the process of biological evolution, and the framework of the new strategy is given also. The effectiveness analysis of the new strategy is discussed by three characteristics of the reproduction operators. The introduction of atavistic evolutionary strategy is highly compatible with the minimum induction pattern, and increases the population diversity to a certain extent. The experimental results show that the new strategy improves the performance of genetic algorithm on convergence time and solution quality.

Keywords: genetic algorithm; atavistic evolutionary strategy; atavistic operator; atavistic probability; premature convergence.

1 Introduction

Genetic algorithm (GA) is a kind of stochastic search algorithm simulating the biological genetics and natural selection mechanism. Through studying the running mechanism of GA, Rudolph proved that GA does not necessarily converge only when reserving the best individuals [1]. From the perspective of mathematical analysis, the convergence process of GA can be viewed as an infinite approximation process, for example, the Markov process. When GA converges, the founded solutions are usually some approximate solutions or satisfactory solutions.

The premature convergence phenomena of GA occur while the population evolves and reaches to a non-global optimal status. Under the status, the better solutions couldn't be found after more iteration operations. Xu etc. [2] put forward the concepts, premature set and population diversity, to analyze the causes and features that result in premature convergence, and illustrated that the maturation effect is the main cause which drives GA into premature convergence. Meanwhile, the premature convergence characteristic shows that the population diversity monotonically decreases. Accordingly, they established a new GA which could converge to the global optimal solutions with probability. Wang etc. [3] gave a new definition of premature convergence based on the fuzzy theory and method to establish the fuzzy model and measurement of population diversity,

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 497–505, 2011.
© Springer-Verlag Berlin Heidelberg 2011

and designed a new strategy to prevent premature convergence in which the crossover and mutation probability could be self-adapt according to the population diversity. The new strategy could improve the performance of GA and prevent premature convergence. Fu etc. [4] put forward the definition of schema coefficient on the basis of the schema theorem to estimate the degree of schema monotonous. The new method, utilizing the schema coefficient to adjust the mutation probability, avoided the premature convergence due to schema monotonous when the algorithm converges. Zhou etc. [5] introduced the definition of life expectancy from the perspective recuperating losing schema and improving schema concentration. The population diversity was kept through controlling the schema diversity. Zhang etc. [6] put forward the ideal density model which guides the population searching toward to the chromosome family whose ancestral individuals are with higher fitness. Sultan etc. [7] introduced the time-table definition to control evolution strategy to keep the population diversity. Hrstka etc. [8] brought forward a targeted improved real encoding to control the premature convergence of GA. Many researchers introduced different encoding, evolutionary strategy, and diversity controlling method and so on, and these new methods prevent premature convergence to some extent. Above-mentioned researches show that the individuals with higher fitness could evolve adequately and the population would prematurely converge if the population does not have adequate diversity.

In general GA, the genetic information will be changed when the number and structure of chromosome change, and this results in that the offspring characteristic changes also. To keep the population evolving and ultimately converging, the population must reserve the individual with best fitness [1]. The population scale must constrain in a certain number due to the constraint of computer storage space and computation time, so the individuals with better fitness replace those with worse fitness under the action of the selection operator, and these individuals will occupy the whole population after some evolution operations. Temporally, the algorithms fall into the premature convergence if it hadn't found the global optimal solution, and couldn't jump out the current search region which is reduced by this schema. There are two premature convergence features, the one is that the algorithm converged in the region not including the global optimal solutions; the other is that the algorithm fluctuated in the vicinity of the global optimal solution but couldn't find the global optimal solution. Fu etc. [9] found that the ability finding optimal solution will be enhanced if adding the atavistic operation into GA. Atavistic operation tries to help GA to break away the status of premature convergence. It might be a new effective method to retrain the premature convergence. Nevertheless, it isn’t given how to implement atavistic operation and related analysis in his work. In this paper, we demonstrate that the atavistic operation can be used to restrain the premature convergence for the *traveling salesman problem* (TSP).

2 Atavistic Evolutionary Strategies

In biology, the population develops from simple to complex, from lower to higher according to the natural selection mechanism. Atavistic phenomena, as a kind of special genetic phenomena, provide the persuasive evidence for biological evolution. Atavism is that some organisms present one or more ancestral properties which had

disappeared for a long time, and is a kind of unusual biological degradation phenomena. When applying the phenomena to GA, the diversity would be increased to some extent, but the difficulty is how to judge which properties belong to atavism, and to control or induct these properties to increase the population diversity. For the above, the atavistic strategy for GA is put forward in this paper. The following briefly describes the process of GA, and then gives atavistic strategy on this basis.

2.1 Operation Process of GA

Assumed that L is the encoding length, H_L is the individual space, R_0^+ is the positive real set, $F:\Omega\rightarrow R_0^+$ is the fitness function. Remarked $\overrightarrow{X}^N(t)\subset H_L$ as t^{th} generation N-order population. The operation process of general GA is as follow.

Algorithm 1: Operation process of general GA

First step (Initialization):

Setting the population scale N, crossover probability P_c, mutation probability P_m and stop condition; stochastically generating N individuals as the initial population $\overrightarrow{X}(0)$; $t\leftarrow 0$.

Second step (Evaluating fitness):

Computing all individual fitness in $\overrightarrow{X}^N(0)$.

Third step (Evolving):

Selecting: Selecting $M/2$ pairs individuals as parents from $\overrightarrow{X}^N(t)$ utilizing selection operator, $M\geq N$.

Crossover: Executing crossover operation on these parents according to P_c, and generating M candidate individuals.

Mutation: Applying mutation operation to these new individuals according to P_m.

Evaluating fitness: Computing all fitness of M new candidate individuals.

Breeding: Selecting N individuals from $\overrightarrow{X}^N(t)$ and M new candidate individuals to constitute new generation population $\overrightarrow{X}^N(t+1)$.

Forth step (Checking stop condition):

If satisfying stop condition, returning the individual with best fitness in $\overrightarrow{X}^N(t+1)$, otherwise setting $t\leftarrow t+1$ and jumping to the third step.

In algorithm 1, the breeding operation in the third step is an important step to control the diversity, and to implement different evolution strategy also. In the process constituting next generation population, we could constitute next generation population whose diversity is more ideal by gene type, gene block, effective alleles, individual similarity and information entropy and so forth.

2.2 Atavistic Evolution Strategy

Assumed that $\overrightarrow{X}^M(t)$ is the t^{th} generation population, $SE: \overrightarrow{X}^M(t) \to \overrightarrow{X}^N(t+1)$ is the breeding operation.

Definition 1 If $\exists x^i(t+1) \in \overrightarrow{X}^N(t+1)$, referring $F[x^i(t+1)] < \min_{\forall x^j(t) \in X^N(t)} F[x^j(t)]$ as *degradation*.

Many researches pointed out that the degradation would restrain the absorption of high-fitness individuals to low-fitness individuals to some extent. It is beneficial to keeping the diversity, and then expanding the search space of GA [9]. The absorption will gradually assimilate the low-fitness individuals under the action of minimal induct schema. This is the root cause for premature phenomena of GA [2]. Above analysis is fit for new individuals, including the individuals generated during evolution, those individuals generated stochastically and those individuals from other populations in multi-population evolution.

Now assumed that the minimal induct schema in population is $L_{\min}(\overrightarrow{X})$, and if an individual $x \in H_L$ includes the minimal induct schema, remarked as $x \propto L_{\min}(\overrightarrow{X})$.

Definition 2 If $\exists x^i(t+1) \in \overrightarrow{X}^N(t+1)$, referring $x^i(t+1) \propto L_{\min}(\overrightarrow{X})$ and $F[x^i(t+1)] < \min_{\forall x^j(t) \in X^N(t)} F[x^j(t)]$ as *atavism*.

We could know that atavism is a special instance of degradation from definition 2. The minimal induct schema affects all individuals in population through-out the process of evolution, so it is generally implied in some individuals, but there isn't an effective model or method to judge it up to now. Therefore, the methods, stochastically generating individuals or selecting individuals from other populations, aren't fit for atavistic operation, because the probability which these new individuals include the minimal induct schema of current population is so low that these new individuals will be absorbed by the schema sooner or later. Although the atavism function could be implemented, the effect would be not very good. In the meanwhile, it will spend a large amount of storage space if reserving the populations through the generations, so the atavism strategy is reserving the initial population as a copy, and the copy provides all chromosomes for the strategy. New atavism strategy is as follow.

1. reserving the initial population as a chromosome copy $\vec{P}$ after initializing the population;
2. Executing the following process when selecting an individual from the candidate population:
   1.1. Adding the individual into next generation population $\overrightarrow{X}^N(t+1)$ if the fitness of the individual is higher or the highest;

1.2. If the individual couldn't be added into next generation according to the atavistic probability, stochastically selecting a individual from $\vec{P}$ and adding into $\overrightarrow{X}^N(t+1)$.

2.3 Atavistic Evolution GA

The framework of new GA with atavistic operation is as algorithm 2.

Algorithm 2: New GA with atavistic operation.

First step (Initialization):

Setting the population scale N, crossover probability P_c, mutation probability P_m, atavistic probability P_a and stop condition; stochastically generating N individuals as initial population $\overrightarrow{X}(0)$; $t \leftarrow 0$; $\vec{P} \leftarrow \overrightarrow{X}(0)$.

Second step (Evaluating fitness):

Computing all individual fitness in $\overrightarrow{X}^N(t)$.

Third step (Evolving):

Selecting: Selecting $M/2$ pairs individuals as parents from $\overrightarrow{X}^N(t)$ utilizing selection operator, $M \geq N$.

Crossover: Executing crossover operation on these parents according to P_c, and generating M candidate individuals.

Mutation: Applying mutation operation to these candidate individuals according to P_m.

Evaluating fitness: Computing all fitness of M new candidate individuals.

Atavistic breeding: Selecting N individuals from M new candidate individuals; Replacing the individuals whose fitness isn't the highest by the individuals selected stochastically from $\vec{P}$ according to P_a; Constituting next generation $\overrightarrow{X}^N(t+1)$.

Forth step (Checking stop condition):

If satisfying stop condition, returning the individual with best fitness in $\overrightarrow{X}^N(t+1)$, otherwise setting $t \leftarrow t+1$ and jumping to the third step.

The following is analyzing the effect of atavism to the algorithm convergence and the population diversity.

3 Effectiveness Analyzing of Atavistic Strategy

Aggregating rate A_E, scattering rate S_E^k and stability rate ST_E^k of abstract operators E in general GA are as follow [10]:

$$\begin{cases} A_E = \min\left\{\{P \mid n[E(\overrightarrow{X})] \geq 1\} : n(\overrightarrow{X}) = 0, \overrightarrow{X} \subset H_L\right\} \\ S_E^k = \max\left\{\{P \mid n[E(\overrightarrow{X})] = 0\} : n(\overrightarrow{X}) \geq k, \overrightarrow{X} \subset H_L\right\} \\ ST_E^k = \min\left\{\{P \mid n[E(\overrightarrow{X})] \geq k\} : n(\overrightarrow{X}) \geq k, \overrightarrow{X} \subset H_L\right\} \end{cases} \tag{1}$$

n represents a kind of selection, crossover or mutation. According to above-mentioned settings, the GA with atavistic operation will add an atavistic operator *SE*. The operator has all of three above properties. In the meanwhile, its transition probability is

$$\begin{cases} P\{SE(\overrightarrow{X})_{ij} = 1 - x_{ij}\} = P_{SE} \\ P\{SE(\overrightarrow{X})_{ij} = x_{ij}\} = 1 - P_{SE} \end{cases} \tag{2}$$

P_{SE} is a predetermined atavistic probability; x_{ij} represents the j^{th} gene in individual X^i. if $P_{SE} \leq 1/2$, the aggregating rate, scattering rate and stability rate of *SE* are

$$\begin{cases} A_{SE} = 1 - (1 - P_{SE}^L)^M \\ S_{SE}^k = [1 - (1 - P_{SE})^L]^k (1 - P_{SE}^L)^{M-k} \\ ST_{SE}^k = (1 - P_{SE})^{Lk} \end{cases} \tag{3}$$

If $P_{SE} \neq 0$, the values of three properties are not zero. It represents that atavistic operator has better anti-absorption and scattering ability. On the premise of reserving the best individual, atavistic operator *SE* changes all locus of a chromosome, so the operator is just like the mutation operator. Therefore, the properties of GA with atavistic operator will change as follow.

$$\begin{cases} A_G = 1 - \left[1 - (P_m + P_{SE})^L\right]^M \\ S_G^M = [1 - (1 - (P_m + P_{SE}))^L]^M \\ ST_G^k \geq \left[\left(1 - \dfrac{(L-1)P_c}{L}\right)^k\right]\left(1 - (P_m + P_{SE})\right)^{Lk} \end{cases} \tag{4}$$

G is a term of the genetic operators as a group. If the GA with atavistic operator had been weak convergence with probability, the GA would be satisfied the following three conditions, because the atavistic operator doesn't change selection, crossover and mutation operators [10].

- The selection pressure of $\{S_t\}$ has a consistent lower bound m_0. There exists a positive integer m_0 makes $P_{S(t)} \geq m_0$.
- The aggregating rate $\{A_{E(t)}\}$ of $\{E_t\}$ is satisfied $\sum_t^{\infty} A_{E(t)} = \infty$.
- Selection intensity $\{\alpha_{S(t)}\}$, aggregating rate $\{A_{E(t)}\}$ and scattering rate $\left\{S_{E(t)}^{m_0}\right\}$ are satisfied the following relationship

$$\lim_{t\to\infty}\frac{1-(1-S_{E(t)}^{m_0})\times\alpha_{E(t)}}{A_{E(t)}}=0 \tag{5}$$

For the above three conditions, the first condition is satisfied because atavistic operator has no effect on selection operator. When $P_m+P_{SE}<1$, the second condition can be apparently met. As $\left[1-(1-(P_m+P_{SE})^L)^M\right]>\left[1-(1-P_m^L)^M\right]$ and $\left[1-\left(1-(P_m+P_{SE})\right)^L\right]^M<\left[1-\left(1-P_m\right)^L\right]^M$, we can know that the GA also meet the third condition. Therefore, the atavistic operator doesn't damage the property of weak convergence with probability.

Similar to the above process, if the original GA satisfies the strong convergence with probability, it will remain strong convergence with probability algorithm after being introduced the atavistic operator. This paper will not repeat them.

4 Experimental Simulation and Analysis

The following experimental environment is Intel T8300 2.4GHz CPU, 2GB RAM, Microsoft Windows XP operating system. The primary parameters of GA are crossover probability 0.615, mutation probability 0.15, atavistic probability 0.025, population scale 50, maximum evolution generation 5 000. All algorithms will stop once finding the global optimal solution (GOS). The evolutionary rules are as follow: if the fitness of an offspring individual is higher than all fitness of parent individuals, replacing the parent individual whose fitness is lower than the fitness of the offspring individual with the offspring individual, and stopping this generation evolution. The local search algorithm is 2-Opt for better observing experimental situation. All experimental datasets are from TSPLIB95 and their scales are less than 100 cities.

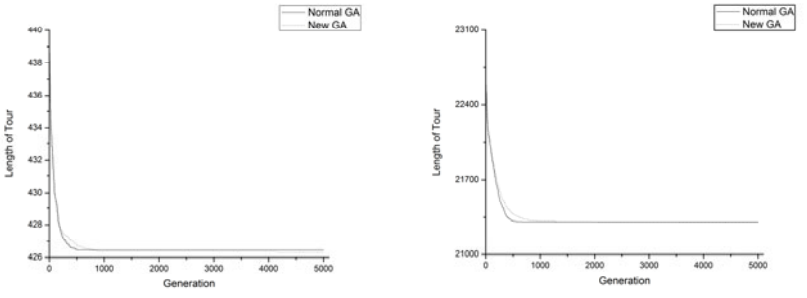
Table 1. Comparison Experiment on atavistic GA

Dataset Name	Cities	Global Optimal Solution	General GA			GA with Atavistic Operator		
			Average Length	Average Time (s)	Times finding GOS	Average Length	Average Time (s)	Times finding GOS
burma14	14	3 323	3 323	0.030	30	3 323	0.033	30
att48	48	10 628	10 628	2.174	30	10 628	1.976	30
eil51	*51*	*426*	*427*	*38.389*	*16*	*427*	*7.765*	*20*
berlin52	52	7 542	7 542	0.159	30	7 542	0.204	30
st70	70	675	675	1.358	30	675	1.463	30
eil76	76	538	538	3.661	30	538	3.178	30
pr76	76	108 159	108 159	1.420	30	108 159	1.558	30
rat99	99	1 211	1 211	2.665	30	1 211	3.221	30
kroa100	100	21 282	21 282	1.793	30	21 282	1.553	30
krob100	*100*	*22 141*	*22 143*	*8.154*	*29*	*22 141*	*4.426*	*30*
kroc100	100	20 749	20 749	2.551	30	20 749	2.641	30
krod100	*100*	*21 294*	*21 306*	*113.485*	*6*	*21 303*	*19.075*	*13*
kroe100	*100*	*22 068*	*22 103*	*91.706*	*10*	*22 099*	*17.478*	*12*
rd100	100	7 910	7 910	2.037	30	7 910	2.146	30

Each algorithm runs on a dataset for 30 times, and computing statistic data. The result is as table1. The convergence times recorded how many times the algorithm converges to the global optimal solution for 30 times repeated running.

We could know from the experimental results:

- For those datasets which could be found its global optimal solutions for 30 times, new GA spent a little more time than the general GA due to the scattering action of the atavistic operation. The convergence analysis figures of dataset eil51 and krob100 are given in figure 1.



(a) eil51 (b) krob100

Fig. 1. Convergence analysis for data set eil51 and krob100 from TSPLIB95

Average convergence speed of new GA is slow than general GA due to the scattering action of atavistic operator at initial phase of algorithm running, but the speed is fast than general GA at later phase of algorithm running because the diversity is better than general GA under the action of atavistic operator.

- For those datasets which can't be found its global optimal solutions for each running, the times which new GA converges to the global optimal solutions is more than the general GA, and average convergence time is significantly shortened.
- No large-scale datasets are used in our experiments because the ability searching optimal solutions of 2-Opt is so weak that it needs a long time to converge to optimal solutions, especially using the atavistic operation. If solving some large-scale datasets, some advanced local search algorithms can be employed, such as 2.5-Opt, 3-Opt or Lin-Kernighan and its variants. While employed these advanced local search algorithms, the problem when and how to select, establish or update the ancestral chromosome library will be discussed further more.
- Atavistic probability must be introduced into the new GA because utilizing atavistic operator. The value of atavistic probability has a greater impact on GA. A larger probability will increase the scattering ability of atavistic operator. This will result in that the convergence speed slows down or GA converges to some worse optimal solutions. A smaller probability will decrease the ability, although this can accelerate search speed and shorten convergence time, but the diversity will be decreased also. Especially, the new GA will degrade to general GA while the value is zero. The suggested value is 0.025 in this paper.

5 Conclusions

This paper brought forward the atavistic strategy for GA according to the inherent atavism phenomena existing in the process of biological evolution. New operation has the property of simplicity and effectiveness. In this paper, the effectiveness of new operation is analyzed through three characteristics of breeding operators in GA, and validated through experiments. Experimental results showed that the new operation can improve the performance of GA from convergence speed to solution quality. It is demonstrated also how to apply the new operator to solving some large-scale traveling salesman problems.

Acknowledgments. This paper is supported by Natural Science Foundation of Guangdong Province (10152800001000029) and Natural Science Foundation of Guangdong Province (10252800001000001).

References

1. Rudolph, G.: Convergence Properties of Canonical Genetic Algorithms. IEEE Transactions on Neural Networks 5, 96–101 (1994)
2. Xu, Z.B., Gao, Y.: Analysis and prevention of the genetic algorithm premature characteristics. Science in China, Series E 26, 364 (1996)
3. Wang, M.L., Wang, X.G., Liu, G.: Quantitative analysis and prevention of genetic algorithm premature convergence. Systems Engineering and Electronics 28, 1249–1251 (2006)
4. Fu, X.H., Kang, L.: Study of the premature convergence of genetic algorithms. Journal of Huazhong University of Science and Technology (Nature Science) 31, 53–54 (2003)
5. Zhou, H.W., Yuan, J.H., Zhang, L.S.: Improved Politics of Genetic Algorithms for Premature. Computer Engineering 33, 201–203 (2007)
6. Zhang, L., Zhang, B.: Research on the Mechanism of Genetic Algorithms. Journal of Software 11, 945–952 (2000)
7. Sultan, B.M., Mahmud, R., Sulaiman, M.N.: Reducing Premature Convergence Problem through Numbers Structuring in Genetic Algorithm. International Journal of Computer Science and Network Security 7, 215–217 (2007)
8. Hrstka, A.L.: Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Advances in Engineering Software 35, 237–246 (2004)
9. Fu, X.F.: An Algebraic Model for State Space of GA. Mathematics in Practice and Theory 35, 119–123 (2005)
10. Xu, Z.B., Zhang, J.S., Zheng, Y.L.: The bionics in computation intelligence: theory and algorithm. Science Press, Beijing (2003)

An Improved Co-Evolution Genetic Algorithm for Combinatorial Optimization Problems

Nan Li and Yi Luo

School of Control and Computer Engineering,
North China Electric Power University, Beijing, China
luna_sliver_blue@yahoo.com.cn,
lyphzh@163.com.cn

Abstract. This paper presents an improved co-evolution genetic algorithm (ICGA), which uses the methodology of game theory to solve the mode deception and premature convergence problem. In ICGA, groups become different players in the game. Mutation operator is designed to simulate the situation in the evolutionary stable strategy. Information transfer mode is added to ICGA to provide greater decision-making space. ICGA is used to solve large-scale deceptive problems and an optimal control problem. Results of numerical tests validate the algorithm's excellent performance.

Keywords: genetic algorithm; co-evolution; deceptive function; game theory; optimal control.

1 Introduction

Since 1980s, researchers in different fields pay more attention to genetic algorithm. Nowadays, genetic algorithm has been made a wide range of applications in many areas.

Holland proposed the building block hypothesis: short, highly fit combinations of bits can combine to form optima. This hypothesis, which provides the mathematics foundation to explain the mechanism of genetic algorithm, makes the quantity of the good schemas increasing exponentially [1]. In the later, Goldberg improved this hypothesis [2].

If the linkage between necessary bit combinations is too weak, in certain types of problems called deceptive problems, genetic algorithms will converge to suboptimal points [3]. Aiming at the weakness of building block hypothesis, Goldberg constructed the deceptive problem. In his opinion, it is very important to use deceptive problems as the test function to study genetic algorithm or other algorithm having the similar search mechanism.

This paper presents an improved co-evolution genetic algorithm (ICGA) .ICGA uses thoughts of co-evolutionary to establish and maintain a number of different subgroups which are cooperative co-evolution and mutual influence [4] [5].There are great similarities between evolutionary game theory and evolutionary algorithm, such as evolutionary stable strategy and replicator dynamics [6]. The cheap talk can be

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 506–513, 2011.
© Springer-Verlag Berlin Heidelberg 2011

used to improve the information transfer mode among the subgroups [8], so ICGA redesigned the mutation operator and information transfer mode among the subgroups by using the ideas of the game theory.

2 The Improved Co-Evolution Genetic Algorithm (ICGA)

2.1 The Traditional Genetic Algorithm (GA)

GA is originated from the computer simulation in the biological system. Simply stated, it's a search procedure which based on the mechanics of natural selection and natural genetics: any initial population can create new individuals which adapt to the environment better, according to the random selection, crossover and mutation operation.

2.2 Mutation

Suppose that a small group of mutants appears in a large population of individuals, all of whom are programmed to play the same incumbent strategy x .Suppose also that the mutants all are programmed to play some other mutant strategy y . Let the share of mutants in the population be ε , where $\varepsilon \in (0,1)$. Pairs of individuals in this biomorphic population are repeatedly drawn at random to play the game, each individual being drawn with equal probability. Hence, if an individual is drawn to play the game, then the probability that the opponent will play the mutant strategy y is ε , and the probability that the opponent will play the incumbent strategy x is $1-\varepsilon$ .The payoff in a match in this biomorphic population is the same as in a match with an individual who plays the mixed strategy $w = \varepsilon y + (1-\varepsilon)x$ .Biological intuition suggests that evolutionary forces select against the mutant strategy if and only if its payoff (fitness) is lower than that of the incumbent strategy.

Let P_m be the mutation rate in ICGA and individuals represented as strategy.

The incumbent strategy is x_j^i .The mutant strategy which bases on the incumbent strategy is as follow:

$$z_j = (b-1)\sum_{i=1}^{n} \frac{1}{b^i} a(i) x_{jl}^i \tag{1}$$

$$a(i) = \begin{cases} 1, & \Pr ob\{a(i)=1\} = \dfrac{1}{n} \\ 0, & \Pr ob\{a(i)=0\} = 1-\dfrac{1}{n} \end{cases} \tag{2}$$

The mixed strategy is as follow:

$$y_{jl}^i = (1-P_m)x_{jl}^i + P_m z_j \tag{3}$$

2.3 Information Transfer Mode

Battle of the sexes is one of a large class of games called coordination games, which share the common feature that the players need to coordinate on one of multiple Nash equilibriums. It is a conflict between a man who wants to go to a prize fight and a woman who wants to go to a ballet. The Battle of the Sexes has two Nash equilibriums, which is iterated dominance equilibrium. One of which is the strategy profile (Prize Fight, Prize Fight). And the strategy profile (Ballet, Ballet) is another Nash equilibrium.

Each of the Nash equilibrium in the Battle of the Sexes is pareto-efficient; no other strategy profile increases the payoff of one player without decreasing that of the other.

When the optimal problem has more than one minimal points, algorithms often converge to one of the points. It is similar with the situation which happens in the coordination game, so I think about using this method in game theory to solve this problem.

Now set the number of subgroups be N and the number of the individual in any subgroup be N_1.The ith subgroup represented as $C_i = \{x_j^i\}$, where x_j^i is the jth element of the ith subgroup， $i = 1, 2, \cdots, N$， $j = 1, 2, \cdots, N_1$.The fitness can be calculated by the formula as follow:

$$Fitness(j) = \frac{N \times X^{j-1}}{\sum_{l=1}^{N} X^{l-1}} \tag{4}$$

Let X be the root of the polynomial equation as follow:

$$\sum_{i=0}^{N-1} (sp - i) \times X^i = 0 \tag{5}$$

Sorting the elements of set C_i from large to small, the probability based on the individual fitness, for $M = \max\{[N_1/\delta], 1\}$, where δ belongs to natural number. Get the sequence $\{m_j^i\}$, for $j = 1, 2, \cdots, M$.

Let the ith subgroup C_i 's information transmission be S_i , let $S_0 = S_N$, $S_{N+1} = S_1$; receiving information set is G_i.We can have the formulas as follow:

$$S_i = \{m_1^i, m_2^i, \cdots, m_M^i\} \tag{6}$$

$$G_i = S_{i-1} \cup S_{i+1} \tag{7}$$

2.4 The Optimal Solution and Nash Equilibrium

If the scale of the problem is very large, the precision of evolutionary algorithm (EA) will be badly affected. So, I think EA should represent as the Data Pre-processing, rather than an optimization procedure. It is important that EA provide a solution set of

large-scale problems, which contains some satisfactory solutions. Expecting to find some favorable modes, I analysis the solution set by the principle of the game theory. In my opinion, one optimization procedure of ICGA represents as one game, the objective function represent as the utility function. The optimal solution, which resolves by ICGA, is the dominant strategy of the game. By Theorem 1, the game must have Nash equilibrium. Let the dominated strategy invade dominant strategy to find the Nash equilibrium.

Theorem 1: Every finite strategic-form game has a mixed-strategy equilibrium [9].

3 Numerical Experiments on Deceptive Problems

This part uses three kinds of small-scale deceptive functions, namely, sub functions to form the two types of large-scale deceptive problems: If all of the input variables of the sub functions are arrange side by side. The deceptive problem is strong-linkage. If the input variable of the sub functions is not neighboring. The deceptive problem is weak-linkage. These problems are used to test ICGA's performance. In these problems, all of input variables take 0 or 1.u represented the number of the input variables taking 1. These three sub functions are as follows:

Three-order deceptive function:

$$f_{deceptive3}(a_1, a_2, a_3) = \begin{cases} 0.9, & u=0 \\ 0.8, & u=1 \\ 0, & u=2 \\ 1, & u=3 \end{cases}. \tag{8}$$

Five-order trap function:

$$f_{trap5}(a_1,a_2,a_3,a_4,a_5) = \begin{cases} 5, & u=5 \\ 4-u, & otherwise \end{cases}. \tag{9}$$

Six-order bipolar function:

$$f_{bipolar6}(a_1, a_2, a_3, a_4, a_5, a_6) = \begin{cases} 0.9, & u=3 \\ 0.8, & u=2 \ \ and \ \ u=4 \\ 0, & u=1 \ \ and \ \ u=5 \\ 1, & u=0 \ \ and \ \ u=6 \end{cases}. \tag{10}$$

3.1 Strong-Linkage Deceptive Functions

The three strong-linkage deceptive functions used as follows:

$$f_1(a) = \sum_{i=1}^{n/3} f_{deceptive3}(a_{3i-2}, a_{3i-1}, a_{3i}) \tag{11}$$

$$f_2(a) = \sum_{i=1}^{n/5} f_{trap5}(a_{5i-4}, a_{5i-3}, a_{5i-2}, a_{5i-1}, a_{5i}) \tag{12}$$

$$f_3(a) = \sum_{i=1}^{n/6} f_{bipolar6}(a_{6i-5}, a_{6i-4}, a_{6i-3}, a_{6i-2}, a_{6i-1}, a_{6i}) \tag{13}$$

Table 1. ICGA's average of function evaluations over 50 independent runs comparing with other algorithms for f1-f3

		ICGA	[9]	[10]	[11]
f1	n=30	2550	842	8500	6510
	n=60	3050	3817	27300	25200
	n=90	3550	9790	57000	45300
f2	n=30	2850	869	14300	7150
	n=60	3150	4088	41250	39620
	n=90	6050	8956	75450	67620
f3	n=30	2800	2098	9000	3150
	n=60	11100	16099	36000	13010
	n=90	21600	50260	45900	24310

Table 2. ICGA's average of function evaluations compared with other algorithms

		ICGA	[9]	[13]
f1	n=60	3050	3817	28807
	n=240	52400	96061	235126
	n=510	268400	516144	—
	n=810	856530	1470082	—
	n=990	1064470	2244465	—
f2	n=100	12100	12514	97746
	n=250	15100	99899	478410
	n=510	50200	541398	—
	n=810	265200	1489900	—
	n=990	479600	2443265	—

Results of numerical tests are given in Table 1. These results, which compared with those in [9], [10] and [11], are the average number of function evaluations over 50 independent runs of ICGA when n = 30, 60, and 90.For f1, f2 and f3, When n=30, the computational cost of ICGA is bigger than MAGA in [9]. In the rest of the case, ICGA outperforms the other methods.

As the scale of the problem increases, the number of local extreme become more and more, leading to the greater difficulty of the problem. To validate ICGA's performance in the large-scale problems, Numerical tests are done: for f1, n increases from 60 to 990; for f2, n increases from 100 to 990. The results are given in Table 2. Compared with methods in [9] and [13], ICGA outperforms the other methods.

3.2 Weak-Linkage Deceptive Functions

The four weak-linkage deceptive functions used as follows:

$$f_4(a) = \sum_{i=1}^{n/3} f_{deceptive3}(a_i, a_{i+n/3}, a_{i+2n/3}) \tag{14}$$

$$f_5(a) = \sum_{i=1}^{n/5} f_{trap5}(a_i, a_{i+n/5}, a_{i+2n/5}, a_{i+3n/5}, a_{i+4n/5}) \tag{15}$$

$$f_6(a) = \sum_{i=1}^{n/6} f_{bipolar6}(a_i, a_{i+n/6}, a_{i+2n/6}, a_{i+3n/6}, a_{i+4n/6}, a_{i+5n/6}) \tag{16}$$

Numerical tests are done: for f4, f5 and f6, n increases from 30 to 210 in steps of 30; these results are the average number of function evaluations over 50 independent runs of ICGA when n takes different values. As we can see in the Table 3, the computational cost of ICGA is smaller than MAGA.

Table 3. ICGA's average of function evaluations over 50 independent runs compared with other algorithms for f4-f6

		ICGA	[9]
f4	n=30	21100	69541
	n=60	104200	929707
	n=90	412400	2851883
	n=210	1656800	47719535
f5	n=30	402800	455201
	n=60	4010400	428605105
	n=90	7017000	—
	n=210	10017000	—
f6	n=30	50600	60111
	n=60	804400	1578582
	n=90	3212800	8027175
	n=210	10036000	284445620

4 ICGA Applied in the Optimal Control

The optimal control is the core of modern control theory. Its main problem is that finding the optimal control strategy to make the given performance meet the minimum or the maximum. Dynamic programming and the maximum principle are the main numerical methods to solve an optimal control problem. Because the traditional methods can't solve the optimization problem of non-differentiability and high morbidity, researchers turn their attention to evolutionary algorithm. In this part, ICGA is applied to a discrete linear system's optimal control problem to test algorithm's performance[14].

4.1 Discrete Linear System's Optimal Control Problem

The discrete linear system as follows：

$$\begin{cases} x_1(k+1) = x_2(k) \\ x_2(k+1) = 2 * x_2(k) - x_1(k) + \dfrac{1}{N^2} u(k) \end{cases}, k = 1, 2, \cdots, N \tag{17}$$

In this problem, the objective function of the performance Index defined as follow:

$$f(x,u) = -x_1(N+1) + \frac{1}{2N} \sum_{k=1}^{N} u^2(k) \tag{18}$$

Theoretical optimization of the discrete linear system is:

$$\min f(x,u) = -\frac{1}{3} + \frac{3N-1}{6N^2} + \frac{1}{2N^3} \sum_{k=1}^{N-1} k^2 \tag{19}$$

4.2 Results

Numerical tests are done: for the discrete linear system, N from 10 to 50 in the steps of 10. The results show the good global searching ability of this new algorithm.

Table 4. Comparison between ICGA and GA

	Algorithms	theory value	Mean of optimal solution	Average termination algebra	Error
N=10	GA	-0.1425	-0.1407	82	0.0018
	ICGA		-0.1424	43	0.0001
N=20	GA	-0.1544	-0.1496	108	0.0048
	ICGA		-0.1541	44	0.0003
N=30	GA	-0.1584	-0.1507	182	0.0077
	ICGA		-0.1582	52	0.0002

5 Conclusion

ICGA is an adaptive, robust optimal algorithm, which base on the thoughts of co-evolutionary and the methodology in game theory .The mutation operation is redesigned and information transfer mode is added. In the experiments, strong-linkage and weak-linkage deceptive functions are used to test the performance of ICGA. The results show algorithms excellent performance. In the end, ICGA applied to an optimal control problem. The results verify the applicability of ICGA.

References

1. Holland, J.: Adaptation in Natural and Artificial Systems. The MIT Press, Cambridge (1992)
2. Goldberg, D.E.: Genetic Algorithm in Search, Optimization and Machine Learning. Addison-Wesley, Massachusetts (1998)

3. Goldberg, D.E.: Messy Genetic Algorithms Revisited Studies in Mixed Size and Scale. Complex Systems 4, 415–444 (1990)
4. Ye, J., Liu, X., Lu, H.: An Evolutionary Algorithm based on Stochastic Weighted Learning for Continuous Optimization. In: 2003 IEEE International Conference on Neural Networks & Signal Processing, Nanjing, China, pp. 14–17 (2003)
5. Liu, J., Zhong, W., Jiao, L., Liu, F.: Multiobjective optimization based on coevolutionary algorithm. In: Tsumoto, S., Słowiński, R., Komorowski, J., Grzymała-Busse, J.W. (eds.) RSCTC 2004. LNCS (LNAI), vol. 3066, pp. 774–779. Springer, Heidelberg (2004)
6. Weibull, J.W.: Evolutionary Game Theory. The MIT Press, Cambridge (1996)
7. Li, Y., Liu, Y., Liu, X.: Active Vibration Control of a Modular Robot Combining a BP Neural Network with a Genetic Algorithm. Journal of Vibration and Control 11(1), 3–17 (2005)
8. Rasmusen, E.: Games and information. Basil Blackwell, Oxford (2006)
9. Liu, J., Zhong, W., Jiao, L.: A multiagent evolutionary algorithm for combinatorial optimization problems. IEEE Trans. on Systems, Man, and Cybernetics, Part B (2009) (in press)
10. Pelikan, M., Goldberg, D.E.: BOA: The Bayesian optimization Algorithm. IlliGAL Report. 98013. Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana-Champaign, Urbana, IL (1998)
11. Lin, Y., Yang, X.: Research on fast evolutionary algorithms based on probabilistic models. Acta Electronica Sinica 29(2), 178–181 (2001) (in Chinese)
12. Wu, S., Zhang, Q., Chen, H.: A new evolutionary algorithm based on family eugenics. J. Softw. 8(2), 137–144 (1997) (in Chinese)
13. Pelikan, M.: Bayesian Optimization Algorithm: From Single Level to Hierarchy. Illinois Genetic Algorithms Lab., Univ. Illinois, Urbana-Champaign, Urbana, IL (2002)
14. Li, Y., Leong, S.H.: Kinematics Control of Redundant Manipulators Using CMAC Neural Network Combined with Genetic Algorithm. Robotica 22(6), 611–621 (2004)

Recursive Structure Element Decomposition Using Migration Fitness Scaling Genetic Algorithm

Yudong Zhang and Lenan Wu*

School of Information Science and Engineering,
Southeast University, Nanjing, China
zhangyudongnuaa@gmail.com, wuln@seu.edu.cn

Abstract. This paper proposed an improved decomposition approach for structuring elements of arbitrary shape. For the model of this method, we use the recursive model which decomposes a given structuring element into a variable-size matrix dilated by a fixed-size matrix and with union of a residue component, such procedures repeated until the variable-size matrix is smaller than a predefined threshold. For the algorithm of our method, we proposed an improved GA based on the ring topology of migration model and the power-rank fitness scaling strategy. The experiments demonstrate that our method is more robust than Park's method, Anelli's method, and Shih's method, and gave the final decomposition tree of different SE shapes such as the letter "V", heart, and umbrella.

Keywords: mathematical morphology, structuring element decomposition, genetic algorithm, migration model, fitness scaling.

1 Introduction

Mathematical morphology (MM) is a theory and technique for the analysis and processing of geometrical structures, based on set theory, lattice theory, topology, and random functions. The basic idea in MM is to probe an image with a simple, predefined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple probe is called structuring element (SE), and is itself a binary image. However, implementation becomes difficult when the size of SE is large [1]. Hence, the techniques for decomposing a large SE into combined small SEs are extremely important.

Many techniques have been proposed for the decomposition of SE, but they are complicated and appear indecomposable cases [2]. Hashimoto indicated that traditional algorithms have the disadvantage of being unable to decompose many simply connected decomposable SEs [3]. Shih [4] developed a method for decomposing arbitrarily shaped binary SEs using canonical genetic algorithm (CGA). The algorithm performs an iterative process to create new ones that minimize a given fitness function. After a sufficient number of iterations, the algorithm tends to converge toward the optimal solution [5].

* Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 514–521, 2011.
© Springer-Verlag Berlin Heidelberg 2011

However, the algorithm suffers from two disadvantages since it takes too much time and the success rate is not satisfying.

In this study, we present an improved genetic algorithm for decomposing arbitrary SEs. The rest of the paper is organized as follows. In Section 2 we introduced the basic principles of SE decomposition. Section 3 describes the recursive model. In Section 4 we transfer the recursive model to an optimization problem which is suitable for GA. In Section 5 we propose a new migration fitness scaling GA. Section 6 compare our method with Park's, Shih's, and Anelli's method. Final Section 7 is devoted to conclusion.

2 Background

Let A denotes a binary image and S denotes a SE. If we decompose S into $S_1 \oplus S_2 \oplus \ldots \oplus S_k$, the dilation and erosion will become as follows according to associative law

$$A \oplus S = A \oplus (S_1 \oplus S_2 \oplus \cdots \oplus S_k) = (((A \oplus S_1) \oplus S_2) \cdots \oplus S_k) \tag{1}$$

$$A - S = A - (S_1 \oplus S_2 \oplus \cdots \oplus S_k) = (((A - S_1) - S_2) \cdots - S_k) \tag{2}$$

The advantage of SE decomposition is that the time complexity for the morphological operations will be reduced extremely. The time complexity for dilation (erosion) operators is proportional to the number of nonzero element of S. Fig. 1(a-b) shows two examples. The first is to decompose a square SE of size 5×5 into two row vectors of size 1×3 and two column vectors of size 3×1, and the computation time will decrease from 25 to 3+2+3+2=10. The second example is to decompose a diamond SE of diameter 7 into three small SEs, and the time will decrease from 25 to 5+4+4=13.

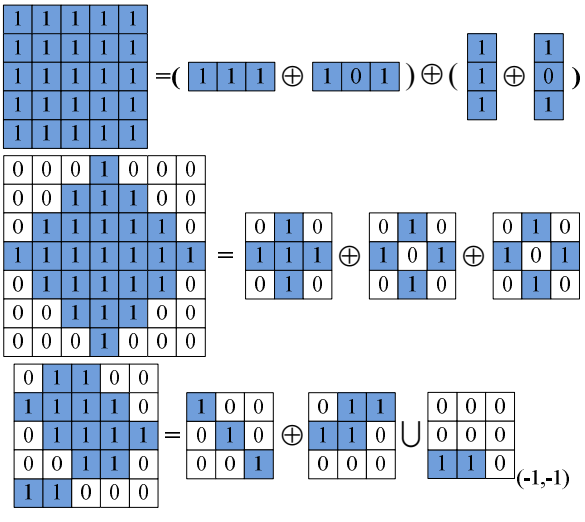


Fig. 1. Examples of SE Decomposition: (a) 7×7 Square; (b) 7×7 Diamond; (c) 5×5 Rand

For parallel pipelined architecture, we decompose the SE using both dilation and union operator. The cost time of the union operator is the maximum of the number of nonzero element of both operands. Fig. 1(c) gives an example, here the subscript (-1,-1) denotes this square matrix should be translated -1 at both *x*-axis and *y*-axis. The parallel computation time after decomposition will decrease from 14 to max(3+4,2) = 7.

3 Recursive Model

Let S_{NN} denote a SE of size $N \times N$. The first decomposition can be written as

$$S_{NN} = V_{(N-2)(N-2)} \oplus F_{33}^{PC} \bigcup R_{NN} \tag{3}$$

Here we use the dilation and union model: the V denotes the variable-size matrix of SE, the F^{PC} denotes the fixed-size prime component, and R denotes the residue component. The R can be easily decomposed into union of factors of size 3×3 because the size 3×3 is often used as the elementary structuring component for decomposition in literature.

$$S_{NN} = V_{(N-2)(N-2)} \oplus F_{33}^{PC} \bigcup \left[\left(R_{33}^{1}\right)_{(x1,y1)} \bigcup \left(R_{33}^{2}\right)_{(x2,y2)} \bigcup \cdots \bigcup \left(R_{33}^{n}\right)_{(xn,yn)} \right] \tag{4}$$

Here n denotes the number of 3-by-3 residual matrix. The subscript (x_i, y_i) denotes that the R^i should be translated by (x_i, y_i). Therefore, the iteration continues to $V_{(N-2)(N-2)}$ until the size is smaller or equal to 3×3. The flow of the recursive decomposition model is depicted in Fig. 2.

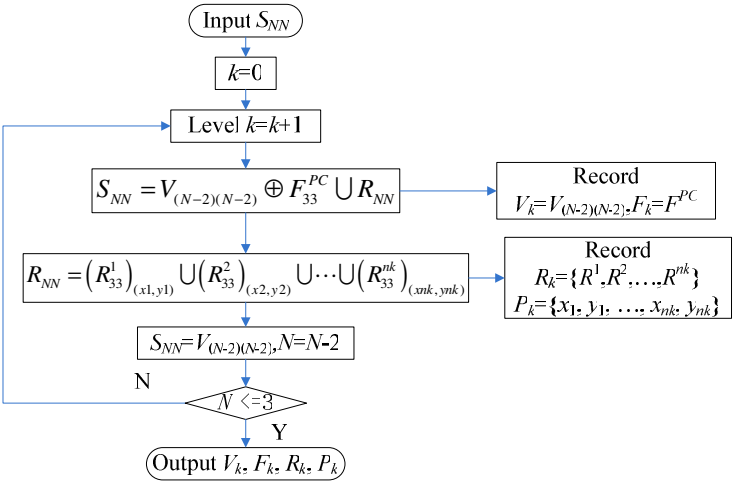


Fig. 2. Flow of Recursive Decomposition Model

Fig. 2 indicates we record 4 variables (V_k, F_k, R_k, and P_k) at each iteration level k, and the decomposition tree can be depicted via those variables. Our task is to develop an effective method to decompose formula (3).

4 Optimization Problem

GA simulates the process of natural selection, and is well suited to optimize solutions over large combinatorial spaces [6]. To apply GA in decomposition model, the encoding strategy and the fitness function should be determined first, which are presented in the following.

For the encoding strategy, we encoded the variable F_k and straighten it into a one-dimensional string of chromosome. For any SE at any iteration, the F_k is a 3-by-3 two-value image, therefore, the chromosome can be written as

$$\xi \triangleq straighten(F) = f_1 f_2 ... f_9 \tag{5}$$

Here ξ denotes the chromosome and f_i denotes the locus. Fig. 3(a) illustrates the numbering scheme for ξ. Two examples are shown in Fig. 3(b-c), and their 1D string of chromosomes are "100101011" and "011001101" respectively.

1	2	3
4	5	6
7	8	9

1	0	0
1	0	1
0	1	1

0	1	1
0	0	1
1	0	1

Fig. 3. (a) Index of gene positions with two examples: (b) "100101011"; (c) "011001101".

For the fitness function, the prime component F is encoded, therefore, variable matrix of SE V and residual matrix R can be obtained through following formulas

$$V = S - F \tag{6}$$

$$R = S - V \oplus F \tag{7}$$

Then, we can extract two different types of costs. One is serial computation cost J_S, and the other is parallel computation cost J_P. Note that those variables satisfy the equality of $S = V \oplus F \bigcup R$, the costs are written as

$$J_S = \sum V + \sum F + \sum R \tag{8}$$

$$J_P = \sum R \tag{9}$$

In this paper, we combine those two costs with a weight factor ω, viz. $J = \omega J_S + (1-\omega) J_P$. For simplicity, ω=0.5.

5 Migration Fitness Scaling GA

GA is efficient and has the ability of jumping out of local minima, but some times it spends excessive time on redundant iteration [7]. To overcome this problem, we proposed a novel migration fitness-scaling GA (MFSGA) which combines the migration model and the fitness scaling strategy.

The migration model takes the idea of separately evolving subpopulations and extends it by adding a means of selectively sharing genetic information between them. Migration may occur in a variety of ways. Two parameters associated with the migration algorithm [8] are extremely important: the migration interval and the migration rate. The migration interval is the number of generations between each migration, and the migration rate is the number of individuals selected for migration.

For each subpopulation in the distributed GA, migration is accomplished as follows. Every migration interval, the best individuals from one subpopulation replace the worst individuals in its neighbor. Individuals that migrate from one subpopulation to another are copied. They are not removed from the source subpopulation. As with multiple elitist selection, migration represents a tradeoff between exploration of new designs and exploitation of highly fit designs which have already been found. The physical relationship between subpopulations imposed by the topology of the distributed system has an effect on this tradeoff as well. The ring topology used for the proposed migration model ensures local communications between subpopulations. The benefit of this design is that migration occurs locally between adjacent populations on the ring. This yields local exploitation of fit designs, while globally the separate subpopulations are free to explore different types of designs independently [9].

Fitness scaling is the other improvement, which converts the raw fitness scores that are returned by the fitness function to values in a range that is suitable for the selection function [10]. The selection function uses the scaled fitness values to select the particles of the next generation. Then, the selection function assigns a higher probability of selection to particles with higher scaled values. There exist bundles of fitness scaling methods. The most common scaling techniques include traditional linear scaling, rank scaling, power scaling, and top scaling.

Among those fitness scaling methods, the power scaling finds a solution nearly the most quickly due to improvement of diversity but it suffers from unstability [11], meanwhile, the rank scaling show stability on different types of tests. Therefore, a new power-rank scaling method was proposed combing both power and rank strategies as follows

$$fit_i = r_i^k / \sum_{i=1}^{N} r_i^k \tag{10}$$

where r_i is the rank of ith individual, N is the number of population. Our strategy contains a three-step process. First, all individuals are sorted to obtain the ranks. Second, powers are computed for exponential values k. Third, the scaled values are normalized by dividing the sum of the scaled values over the entire population.

6 Experiments

The experiments were carried out on the platform of P4 IBM with 3GHz processor and 2GB memory, running under Windows XP operating system. The algorithm was developed via the global optimization toolbox of Matlab 2010b.

The SE shown in Fig. 4 is indecomposable by Park's algorithm [3]. However, a successful decomposition can be found via our algorithm as shown in Fig. 4(a). The serial computation cost is 10 and the parallel computation cost is 1.

We run both our proposed improved GA and the CGA in Shih's method 20 times [4] without using Park's 13 prime factor as initialization population. Our method all finds the global best result, while three runs of CGA finds a failed result as shown in Fig. 4(b) with serial computation cost of 11 and parallel computation cost of 2. Therefore, the proposed improved GA is more robust than the CGA algorithm of Shih's algorithm.

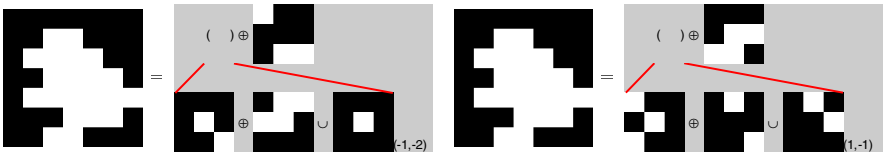


Fig. 4. A typical decomposition tree: (a) A success run; (b) A failed run

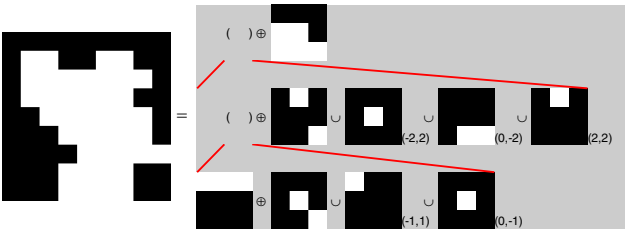
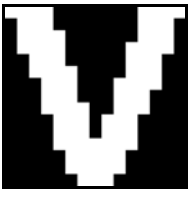
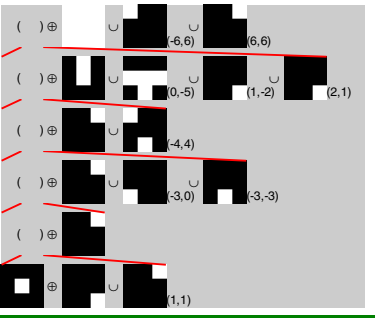
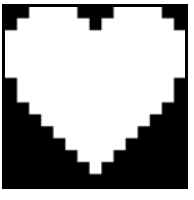
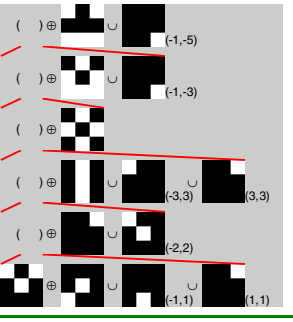
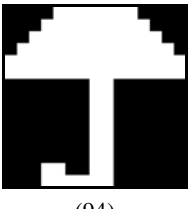
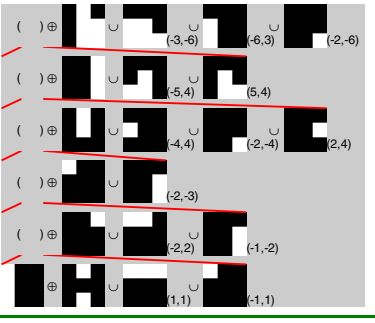


Fig. 5. Decomposition tree of Anelli's SE

Fig. 5 shows the decomposition tree of Anelli's SE. We compared our method with Anelli's paper [12]. The serial computation cost of our algorithm is 18, and the parallel computation cost of our algorithm is 6. Conversely, the serial and parallel computation costs of Anelli's method are 22 and 10, respectively. The decomposition tree of Anelli can be seen in Ref. [12].

We use 4 examples of 15-by-15 different shapes including the letter "V", heart, and umbrella. Tab. 1 lists their original SE shape, the decomposition tree, the serial computation cost, and the parallel computation cost. For the V-shaped SE, the original points are 94, the serial and parallel computation cost after decomposition is only 94 and 13, respectively. For the heart, the points of original SE, J_S, and J_P are 142, 32, and 8, respectively. For the umbrella, the points of original SE, J_S, and J_P are 94, 44, and 27, respectively.

Table 1. Three Decomposition Examples (black denotes 0 & white denotes 1)

Original SE & its points	Decomposition Tree	J_S	J_P
 (94)		29	13
 (142)		32	8
 (94)		44	27

7 Conclusions

In this paper, a decomposition method for arbitrarily-shaped structuring elements is proposed based on recursive model and migration fitness scaling genetic algorithm. The MFSGA method uses the ring topology of the migration model and the power-rank scaling strategy. Compared to Park's method [3], Anellie's method [12], and Shih's method [4], our method is more robust and have higher rate to find global minima. The future work will focus on applying the proposed MFSGA method to various industrial fields including image classification [13], pattern recognition [14], and weights optimization [15].

References

1. Burgeth, B., Bruhn, A., Papenberg, N., Welk, M., Weickert, J.: Mathematical morphology for matrix fields induced by the Loewner ordering in higher dimensions. Signal Processing 87, 277–290 (2007)
2. Park, H., Yoo, J.: Structuring element decomposition for efficient implementation of morphological filters. In: IEE Proceedings of Vision, Image and Signal Processing, vol. 148, pp. 31–35 (2001)
3. Hashimoto, R.F., Barrera, J.: A note on Park and Chin's algorithm [structuring element decomposition]. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 139–144 (2002)
4. Shih, F.Y., Wu, Y.-T.: Decomposition of binary morphological structuring elements based on genetic algorithms. Computer Vision and Image Understanding 99, 291–302 (2005)
5. Zhang, Y., Wu, L.: Stock Market Prediction of S&P 500 via combination of improved BCO Approach and BP Neural Network. Expert Systems with Applications 36, 8849–8854 (2009)
6. Zhang, Y., Yan, J., Wei, G., Wu, L.: Find multi-objective paths in stochastic networks via chaotic immune PSO. Expert Systems with Applications 37, 1911–1919 (2010)
7. Rausch, T., Thomas, A., Camp, N.J., Cannon-Albright, L.A., Facelli, J.C.: A parallel genetic algorithm to discover patterns in genetic markers that indicate predisposition to multifactorial disease. Computers in Biology and Medicine 38, 826–836 (2008)
8. Rom, W.O., Slotnick, S.A.: Order acceptance using genetic algorithms. Computers & Operations Research 36, 1758–1767 (2009)
9. Omara, F.A., Arafa, M.M.: Genetic algorithms for task scheduling problem. Journal of Parallel and Distributed Computing 70, 13–22 (2010)
10. Tkaczyk, E.R., Mauring, K., Tkaczyk, A.H., Krasnenko, V., Ye, J.Y., Baker Jr, J.R., Norris, T.B.: Control of the blue fluorescent protein with advanced evolutionary pulse shaping. Biochemical and Biophysical Research Communications 376, 733–737 (2008)
11. Korsunsky, A.M., Constantinescu, A.: Work of indentation approach to the analysis of hardness and modulus of thin coatings. Materials Science and Engineering: A 423, 28–35 (2006)
12. Anelli, G., Broggi, A., Destri, G.: Decomposition of arbitrarily shaped binary morphological structuring elements using genetic algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 217–224 (1998)
13. Zhang, Y., Wang, S., Wu, L.: A Novel Method for Magnetic Resonance Brain Image Classification based on Adaptive Chaotic PSO. Progress in Electromagnetics Research 109, 325–343 (2010)
14. Zhang, Y., Wu, L.: Pattern recognition via PCNN and Tsallis entropy. Sensors 8, 7518–7529 (2008)
15. Zhang, Y., Wu, L.: Weights optimization of neural network via improved BCO approach. Progress in Electromagnetics Research 83, 185–198 (2008)

A Shadow Price Guided Genetic Algorithm for Energy Aware Task Scheduling on Cloud Computers

Gang Shen and Yan-Qing Zhang

Department of Computer Science, Georgia State University
P.O. Box 3994, Atlanta, GA 30302-3994, USA
gshen1@student.gsu.edu,
yzhang@cs.gsu.edu

Abstract. Minimizing computing energy consumption has many benefits, such as environment protection, cost savings, etc. An important research problem is the energy aware task scheduling for cloud computing. For many diverse computers in a typical cloud computing system, great energy reduction can be achieved by smart optimization methods. The objective of energy aware task scheduling is to efficiently complete all assigned tasks to minimize energy consumption with various constraints. Genetic Algorithm (GA) is a popular and effective optimization algorithm. However, it is much slower than other traditional search algorithms such as heuristic algorithm. In this paper, we propose a shadow price guided algorithm (SGA) to improve the performance of energy aware task scheduling. Experiment results have shown that our energy aware task scheduling algorithm using the new SGA is more effective and faster than the standard GA.

Keywords: Energy Aware Task Scheduling, Genetic Algorithm, Green Computing, Cloud Computing, Optimization, Shadow Price.

1 Introduction

Computer has significantly boosted modern technology development and revolutionized people's lifestyle. Energy used by computers is roughly equivalent to that in aviation industry. It accounts for 2% of anthropogenic CO2 from its share of energy consumption [8].

Cloud computing needs huge amount of energy [9][10]. It is estimated that US servers and data centers consumed about 61 billion kilowatt-hours (kWh) in 2006 (1.5 percent of total U.S. electricity consumption) for a total electricity cost of about $4.5 billion [7]. To protect environment, it is necessary to schedule tasks more efficient for cloud computing. The objective is to use the least amount of energy to complete computing tasks under various constraints.

The energy aware task scheduling methodologies can be categorized as heuristic algorithms [10][17][19], bio-inspired search algorithms [13][16], and hybrid algorithms [14][15].

Heuristic algorithms can find good solutions among all possible ones, but they do not guarantee that the best or the near optimal solution will be found. Bio-inspired search

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 522–529, 2011.
© Springer-Verlag Berlin Heidelberg 2011

algorithms find excellent solutions by simulating nature. The typical algorithms are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), etc. They can find optimal or near optimal solutions. They are less efficient than heuristic algorithms.

This paper uses the shadow price based GA [4][11][12] to solve the energy aware task scheduling problem.

2 Energy Aware Task Scheduling

The amount of power a processor consumes is the product of voltage and current it draws. Increasing voltage can increase the processor's speed. Thus, the power consumption of a processor is directly linked to its running speed.

It is a combinatorial optimization problem to achieve the most efficient balance between the processor voltage and its running speed for a target problem. Li established the theories on efficient energy aware task scheduling for multi-processor or multi-computer [1]. The problem can be defined as:

n cloud computers in a cloud computing system are used to finish *m* tasks by the deadline time *T*. Assume that m_i tasks P_k^i for *k*=1, 2, …, m_i are executed on computer *i* for $m = \sum_{i=1}^{n} m_i$. A changeable speed for task P_k^i is denoted as S_k^i for *i*=1, 2, …, *n*, and *k*=1, 2, …, m_i. The speed is defined as the number of instructions per second. The number of instructions of task P_k^i is denoted as R_k^i. The execution time for task P_k^i on computer *i* is $\frac{R_k^i}{S_k^i}$. The total execution time for m_i tasks P_k^i on computer *i* is defined as $T_i = \sum_{k=1}^{m_i} \frac{R_k^i}{S_k^i}$. According to [1], the energy for P_k^i on computer *i* is $E_k^i = C_i R_k^i [S_k^i]^{\alpha_i - 1}$ where C_i is a constant, $\alpha_i = 1 + \frac{2}{\phi_i} \geq 3$ for $0 < \phi_i \leq 1$, *i*=1, 2, …, *n*, and *k*=1, 2, …, m_i. The total energy is $E = \sum_{i=1}^{n} \sum_{k=1}^{m_i} C_i R_k^i [S_k^i]^{\alpha_i - 1}$.

The optimization problem is [19]:

Minimize $E = \sum_{i=1}^{n} \sum_{k=1}^{m_i} C_i R_k^i [S_k^i]^{\alpha_i - 1}$ with constraints: $1 \leq m_i \leq m - n + 1$, $m = \sum_{i=1}^{n} m_i$, $m > n$, $\sum_{k=1}^{m_i} \frac{R_k^i}{S_k^i} \leq T$ and $a_i \leq S_k^i \leq b_i$ where a_i is the minimum speed and b_i is the maximum speed of computer *i*, respectively, for *i*=1, 2, …, *n*, and *k*=1, 2, …, m_i.

The goal is to design a new energy aware task scheduling algorithm that can find an optimal or near optimal schedule to compete all *m* tasks on *n* computers using minimum or near minimum energy *E* by the deadline time *T*. It's a complex integer combinatorial optimization problem. It is a NP hard problem.

3 Shadow Price Guided Genetic Task Scheduling Algorithm

Genetic Algorithm [2][3] is a reward based multi solution search algorithm. It is a branch of bio inspired evolution algorithm (EA). It has been applied to many applications successfully [18].

There are primary three operations in GA. The crossover operator mimics two parents producing a child in nature. The child shall have some characteristics of both parent and different from both parents. The unary mutation operator modifies the component(s) of a solution. Often times, the newly generated solution is much different than the original and may not even be valid. New solutions are randomly generated to further broaden the search space.

There are two major challenges in the GA search process, solution quality and search speed. All three GA operations heavily rely on randomness. Although the randomness is necessary in GA, it introduces many unnecessary and worse solutions. It slows down the search process and results low quality solution.

3.1 Shadow Price

GA operates on the components of the solution(s). It uses randomness to select component(s) to participate operation and evolve them into random directions. The result can be highly diverse. This directly impacts the algorithm's results and generates a lot of unnecessary calculations.

To add some intelligence into GA operations, a method is needed to evaluate and compare components. Without a unified means to evaluate components, it is difficult for GA operators to select and work on high potential component(s) for generating better solutions. Since the fitness function can only evaluate the overall solution, GA does not provide a method to intelligently select components and guide evolution. Our research did not locate any direct attempt to address this challenge.

We define the shadow price in GA as the relative potential improvement to the solution's (chromosome) fitness value with a change of a component (gene). It's a relative potential improvement since the concept is defined on a single component and a component change may force other components' change to maintain solution's feasibility. The improvement may or may not be realizable. A change of component states the fact that component change can be a value change, a position change, or other applicable changes. We use shadow price to directly compare components, their relationships, and their contributions toward a better solution.

Based on different problems, shadow prices can take on different meanings or values. In the traveling salesman problem, it can simply be the possible distance reductions from changing the next visiting city [4]. In manufacture, shadow price can be the cost of material, time, etc. [11][12]. The definition has to be clear and comparable among components. Here are a few guidelines for selecting shadow price.

- The shadow price shall enable comparison among components. Precise values are preferred over fuzzy values.
- The shadow price shall reflect the attribute of a component such as price, cost, material, etc. The attribute shall directly or indirectly impact the solution quality (fitness value).

- The shadow price for the solution (sum of shadow prices from all components) shall change with the quality of the solution (fitness value). There is no need to define a math function to associate them. The only requirement is to ensure that the shadow price is consistent with the search process.
- The shadow price calculation shall be simple and fast.

There is no direct relationship between a solution's fitness value and its components' shadow prices. The fitness value represents the current solution's position in the search space. The shadow prices represent potential improvements and directions to evolve. The shadow prices are only meaningful in the search process.

3.2 Shadow Price Guided Genetic Algorithm

The objectives of our new shadow price guided GA are to find better solutions and use less time than the standard GA. When search time is not restricted, our new SGA shall find optimal solutions faster. When search time is limited, our new SGA shall find better solutions. We use shadow price to generate many better solutions and reduce the amount of unnecessary calculations. The new SGA can find better solution and find it fast.

We establish a two-measurement system in our new SGA: fitness values are used to evaluate overall solutions and shadow prices are used to evaluate components. We use shadow prices to guide the evolution operations.

The goal of mutation is to improve solution by changing a component's state. Selecting component is the key. Changing a component from a good state to a less quality state is counterproductive. Standard GA provides no help. We use shadow price to define the potential contribution of a component. The component with high potential has high shadow price. We can intelligently select a component with high shadow price to mutate and try to change its state to a lower shadow priced state. That is, we try to realize the component's potential. In order to avoid local optimal trap, component is selected from a pool of high shadow priced components. The new mutation increases the chance of generating better solutions and reduces calculations.

The crossover operator tries to generate a better child by inheriting components from both parents. This can only happen when good attributes are passed down. Standard GA relies on randomness and generates a lot of calculations. We use shadow price to evaluate components' quality and try to pass good components (low shadow priced) to the child. To avoid local trap, randomness are also used to select good components from a pool of low shadow priced components.

Our new SGA still uses the standard GA's framework. We enhance the GA operators with shadow price information. Randomness is used to ensure a global search and avoid local optimal trap. The contribution is that we use shadow price to influence the GA operators to increase the odds of generating better solutions.

3.3 Energy Aware Shadow Price Guided Genetic Task Scheduling Algorithm

There are two optimizations in the energy aware task scheduling problem, assigning tasks to processors and minimizing each processor's energy. Li proved that the total energy consumption is minimized when all tasks are executed with the same power (speed) on a uniprocessor computer [1] (theorem 3). This solves the second optimization

sub problem. In our new algorithm (as shown below), all assigned tasks for a processor are combined to calculate the minimal energy consumption.

Begin

1. Validate there is at least one feasible solution.
2. Build initial population.
3. While stop criteria has not met
   3.1 Select a sub population to randomly apply one of the following operations
   - Classic mutation operation (Move). Randomly select two processors and move one randomly selected task from one processor the other.
   - Classic mutation operation (Exchange). Exchange two randomly selected tasks between two randomly selected processors.
   - Shadow priced guided mutation operation (Move).
     (a) Calculate shadow prices for all processors.
     (b) Establish a pool of high shadow priced processors and random select one processor (Pa).
     (c) Establish a pool of low shadow priced processors and random select one processor (Pb).
     (d) Random select one task from Pa and move it to Pb.
   - Shadow priced guided mutation operation (Exchange).
     (a) Calculate shadow prices for all processors.
     (b) Establish a pool of high shadow priced processors and random select one processor (Pa).
     (c) Establish a pool of low shadow priced processors and random select one processor (Pb).
     (d) Sort Pa and Pb's tasks based on their instruction count.
     (e) Establish a task pool from Pa's tasks whose instruction count are more than average and random select one task.
     (f) Establish a task pool from Pb's tasks whose instruction count are less than average and random select one task.
     (g) Exchange the selected tasks between Pa and Pb.

   3.2 Add random solutions
   3.3 Filter and build next generation
   End While

End

The shadow price SP^i is the average energy consumption per instruction for the processor i. The evolution direction is to reduce processor's shadow price by moving tasks among them. The four mutation operations have equal opportunities to be used when algorithm starts and the odds change with the search progression. The two new shadow price guided operations can greatly improve the GA's performance.

$$SP^i = \frac{\sum_{k=1}^{m_i} E_k^i}{\sum_{k=1}^{m_i} P_k^i}$$

4 Experiment

To evaluate our new algorithm, we conducted a comparative study between GA and our new shadow price guided SGA. Both algorithms followed the same framework and were identical except the mutation operations. SGA used all four mutation operations and standard GA used two. Same calculations were used to optimize the processor's power consumption after tasks have been assigned.

We implemented both algorithms in Microsoft C#. Tests were run on a Lenovo Thinkpad T410 with Intel Core i5-M520 2.4 GHz CPU and 4 GB of memory running Windows 7. Each test was run at least 10 times. Results were averaged and reported.

We selected the latest commercial released CPUs [5] for our experiments. We used published speed as the processor's minimum speed. A random number between 5% and 25% was used as the speed improvement to define the processor's maximum speed. Constants C and Φ were randomly generated. Instruction counts for tasks were also randomly generated between 500 and 100,000. To improve quality, we used a public available true random number generating services [6] instead of C# library.

Table 1. Energy Saving using SGA (*Ega-Esga*)

Cp	Gmax	Ct=500	Ct=1000	Ct=1500	Ct=2000	Ct=3000	Ct=5000
10	500	3.90E+19	7.31E+20	2.10E+21	1.59E+21	1.40E+21	4.00E+21
10	1000	4.90E+15	6.05E+19	8.59E+20	8.00E+20	7.51E+20	2.06E+21
10	2000	4.51E+15	1.46E+16	2.10E+18	1.46E+20	3.65E+20	1.21E+21
20	500	2.22E+19	6.29E+20	4.81E+21	6.41E+21	9.28E+20	1.17E+21
20	1000	6.79E+18	1.42E+20	6.35E+20	1.22E+21	3.27E+20	1.07E+21
20	2000	8.00E+16	2.05E+19	9.83E+19	1.54E+20	2.20E+20	8.36E+20
30	500	5.97E+18	4.85E+19	1.31E+20	1.48E+20	7.94E+19	1.65E+20
30	1000	1.37E+18	1.76E+19	4.85E+19	5.17E+19	4.82E+19	1.35E+20
30	2000	1.24E+17	3.85E+18	1.60E+19	1.89E+19	2.24E+19	6.44E+19
40	500	2.65E+18	1.71E+19	7.28E+19	7.36E+19	3.22E+19	2.85E+19
40	1000	8.39E+17	8.20E+18	2.48E+19	3.30E+19	2.74E+19	5.96E+19
40	2000	9.97E+16	1.58E+18	6.73E+18	8.71E+18	1.34E+19	3.85E+19
50	500	1.60E+18	1.06E+19	1.97E+19	2.02E+19	1.39E+19	1.34E+19
50	1000	3.58E+17	3.45E+18	8.18E+18	8.74E+18	1.18E+19	1.77E+19
50	2000	5.15E+16	8.74E+17	3.07E+18	3.62E+18	4.53E+18	1.59E+19
Average		5.41E+18	1.13E+20	5.88E+20	7.12E+20	2.83E+20	7.25E+20

Experiment cases were created using different combination of processors and tasks. Time constraint for each experiment case was randomly created, validated and shortened to ensure that very few processors can be idle in the optimal solutions.

The first test was to compare solution quality between two algorithms. Each test case is a combination of CPU counts (*Cp*), task counts (*Ct*), and max generations (*Gmax*). Table 1 lists the SGA energy (*Esga*) savings over GA (*Ega*). It is *Ega-Esga*. A positive value states SGA used less energy than GA. In all test cases, Table 1 shows SGA used less energy than standard GA. SGA achieved better solutions.

Next, we conducted speed test between the two algorithms. For each test case, we used the average energy consumption from above test as the stopping criteria. Algorithm only stops when the solution is equal or better than the target energy value.

Table 2. SGA Speed Improvement in Generations, *Gga-Gsga*

Cp	Ct=500	Ct=1000	Ct=1500	Ct=2000	Ct=3000	Ct=5000
10	230	214	221	284	305	217
20	291	145	120	141	221	190
30	333	258	207	241	252	211
40	313	228	223	234	282	306
50	351	260	295	202	231	260
Average	303.6	221	213.2	220.4	258.2	236.8

Table 2 lists the generation savings from SGA (*Gsga*) over GA (*Gga*), *Gga-Gsga.* All positive values show SGA used fewer generations than GA. Table 3 lists the time savings from SGA (*Tsga*) over GA (*Tga*), *Tga-Tsga.* Again, all positive values show SGA is faster than GA to reach the target. Both tables show SGA is faster than GA.

Table 3. SGA Speed Improvement in Time (second), *Tga-Tsga*

Cp	Ct=500	Ct=1000	Ct=1500	Ct=2000	Ct=3000	Ct=5000
10	0.463	1.066	2.313	3.826	7.836	14.876
20	0.561	0.626	0.812	1.292	3.357	7.872
30	0.573	0.778	1.166	1.799	3.458	5.873
40	0.584	0.799	0.882	1.71	2.626	6.296
50	0.681	0.804	1.302	1.209	2.28	4.468
Average	0.5724	0.8146	1.295	1.9672	3.9114	7.877

Our experiment results demonstrated that our new shadow price guided GA can find better solutions than standard GA. SGA is also faster than GA. The experiments validated our design and the effectiveness of our new algorithm.

5 Conclusion

GA is an effective global search algorithm and has been used widely in solving many optimization problems. But GA can take a very long time to solve large complex problems and may provide sub optimal solutions.

We propose a new shadow price guided GA in this paper to improve GA's performance. We add a new component measurement method to the search. We propose a two-measurement system: fitness values are used to evaluate overall solutions and shadow prices are used to evaluate components. We use shadow prices to guide the evolution operations. Our new SGA improves the result qualityand the search speed.

We applied our new SGA to solve the energy aware task scheduling problem and delivered very good results. Experiments showed that our new algorithm achieved better results than the standard GA and used less time.

References

1. Li, K.: Performance Analysis of Power-Aware Task Scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed. IEEE Trans. on Parallel and Distributed Systems 19(11), 1484–1497 (2008), doi:10.1109/TPDS.2008.122

2. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
3. Holland, J.H.: daptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press, Cambridge (1992)
4. Shen, G., Zhang, Y.Q.: A New Evolutionary Algorithm Using Shadow Price Guided Operators. Applied Soft Computing 11(2), 1983–1992 (2011)
5. Wikipedia, Instructions Per Second (2010), http://en.wikipedia.org/wiki/Instructions_per_second (accessed October 2010)
6. Random.org (2010), http://www.random.org (accessed October 2010)
7. US Environmental Protection Agency. EPA Report on Server and Data Center Energy Efficiency (August 2007), http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf (accessed October 2010)
8. Consortium for School Networking Initiative, Some Facts About Computer Energy Use (2010), http://www.cosn.org/Initiatives/GreenComputing/InterestingFacts/tabid/4639/Default.asp (accessed October 2010)
9. von Laszewski, G., Wang, L., Younge, A.J., He, X.: Power-aware scheduling of virtual machines in DVFS-enabled clusters. In: IEEE Intl. Conf. on Cluster Computing and Workshops, 2009, pp. 1–10 (2009), doi:10.1109/CLUSTR.2009.5289182
10. Wang, L., von Laszewski, G., Dayal, J., He, X., Furlani, T.R.: Thermal Aware Workload Scheduling with Backfilling for Green Data Centers. In: The 28th IEEE Intl. Conf. on Performance Computing and Communications (December 2009), doi:10.1109/PCCC.2009.5403821
11. Shen, G., Zhang, Y.Q.: A Novel Genetic Algorithm. In: The 9th International FLINS Conf. on Foundations and Applications of Computational Intelligence (FLINS 2010) (2010)
12. Shen, G., Zhang, Y.Q.: Solving the Stock Reduction Problem with the Genetic Linear Programming Algorithm. In: The 2010 International Conference on Computational and Information Sciences, ICCIS 2010 (2010)
13. Tian, L., Arslan, T.: A genetic algorithm for energy efficient device scheduling in real-time systems. In: 2003 Congress on Evolutionary Computation, pp. 242–247 (2003)
14. Miao, L., Qi, Y., Hou, D., Dai, Y.H., Shi, Y.: A multi-objective hybrid genetic algorithm for energy saving task scheduling in CMP system. In: IEEE Intl. Conf. on Systems, Man and Cybernetics, 2008, pp. 197–201 (2008), doi:10.1109/ICSMC.2008.4811274
15. Liu, Y., Yang, H., Luo, R., Wang, H.: Combining Genetic Algorithms Based Task Mapping and Optimal Voltage Selection for Energy-Efficient Distributed System Synthesis. In: 2006 Intl. Conf. on Communications, Circuits and System, vol. 3, pp. 2074–2078 (2006), doi:10.1109/ICCCAS.2006.285087
16. Chang, P.C., Wu, I.W., Shann, J.J., Chung, C.P.: ETAHM: An energy-aware task allocation algorithm for heterogeneous multiprocessor. In: 45th ACM/IEEE Design Automation Conference, 2008, pp. 776–779 (2008)
17. Li, Y., Liu, Y., Qian, D.: A Heuristic Energy-aware Scheduling Algorithm for Heterogeneous Clusters. In: 2009 15th Intl. Conf. on Parallel and Distributed Systems (ICPADS), pp. 407–413 (2009), doi:10.1109/ICPADS.2009.33
18. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Springer, Heidelberg (2005)
19. Zhang, L.M., Li, K., Zhang, Y.Q.: Green Task Scheduling Algorithms with Speeds Optimization on Heterogeneous Cloud Servers. In: 2010 IEEE/ACM Intl. Conf. on Green Computing and Communications (GreenCom 2010), pp. 76–80 (2010)

A Solution to Bipartite Drawing Problem Using Genetic Algorithm

Salabat Khan, Mohsin Bilal, Muhammad Sharif, and Farrukh Aslam Khan

Department of Computer Science,
National University of Computer and Emerging Sciences,
A.K. Brohi Road, H-11/4, Islamabad, Pakistan
farrukh.aslam@nu.edu.pk

Abstract. Crossing minimization problem in a bipartite graph is a well-known NP-Complete problem. Drawing the directed/undirected graphs such that they are easy to understand and remember requires some drawing aesthetics and crossing minimization is one of them. In this paper, we investigate an intelligent evolutionary technique i.e. Genetic Algorithm (GA) for bipartite drawing problem (BDP). Two techniques GA1 and GA2 are proposed based on Genetic Algorithm. It is shown that these techniques outperform previously known heuristics e.g., MinSort (M-Sort) and BaryCenter (BC) as well as a genetic algorithm based level permutation problem (LPP), especially when applied to low density graphs. The solution is tested over various parameter values of genetic bipartite drawing problem. Experimental results show the promising capability of the proposed solution over previously known heuristics.

Keywords: Crossing Minimization, Bipartite Graph, Genetic Algorithm, Bipartite Drawing Problem (BDP), Crossing Minimization Heuristics (CMH)

1 Introduction

In the context of graph theory, bipartite graph 'G' is a graph whose set of vertices can be partitioned into two non-empty sets or layers; e.g., 'V1' & 'V2' such that the edges only connect the vertices in different layers. Alternatively, 'V1' and 'V2' are also known as the upper and the lower layers of bipartite graph respectively. The edge connectivity may introduce crossings in the bipartite graph. Drawing a graph with as few edge crossings as possible is a well-known NP-Complete problem of graph theory [1, 2, 3, 4]. A more detailed description about bipartite graph may be found in [5, 8, 10, 15]. Watkins [8] restricted the edge crossing minimization problem to the bipartite graphs. Crossing minimization in a bipartite graph has several applications in various graph drawing systems [9], most notably in those based on the Sugiyama's algorithm [11] and is also used in VLSI design [10], bi-clustering [12], networking, and information engineering. Stallmann et al. [15] described the importance of crossing minimization based on two motivational factors; a) it improves the appearance of a graph, and b) it reduces the wiring congestion and crosstalk in VLSI circuits, which in turn may reduce the total wire length and the layout area.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 530–538, 2011.
© Springer-Verlag Berlin Heidelberg 2011

There are two main problems discussed regarding the bipartite graphs. First, the level permutation problem (LPP) where the order of nodes in one layer is kept fixed, and second, the bipartite drawing problem (BDP) where both layers of the bipartite graph can be subject to permutation. In both of these problems, the objective is to minimize the number of crossings in the bipartite graph. Researchers have carried out a fair amount of work on this problem including Catarci [13], Camel and Irene [14], Stallmann et al. [15] and Zheng et al. [16]. Zoheir Ezziane [17] has used an Evolutionary Algorithm (EA) for BDP and compared the results with BaryCenter (BC). There is no crossover operator used in [17] and the experiments are done over very small sizes (10-17) of the bipartite graph. In Laguna et al. [18], the application of tabu search for Arc crossing minimization in hierarchal digraphs is analyzed. Martí [19] used GRASP (Greedy Randomized Adaptive Search Procedure) for Arc crossing minimization problem. In most of these approaches, researchers have tried to permute single layer of the bipartite graph (keeping the second layer fixed i.e. LPP) with the condition that the bipartite graph will have same cardinality of vertices in both layers.

In this paper, we use Genetic Algorithm (GA) for the bipartite drawing problem and propose two techniques i.e., GA1 and GA2. It is shown that these techniques outperform previously known heuristics such as MinSort (M-Sort) [12] and BaryCenter (BC) [11] as well as a Genetic Algorithm based level permutation problem (LPP) especially when applied to low density graphs. In our techniques, we use the X_Order1 crossover operator and experiments are carried over large sizes (15-100) of the bipartite graph that gives a better estimation of the approach based on its performance and scalability. Moreover, both the layers of the bipartite graph are considered to be rearranged during searching or learning, so it is essentially the bipartite drawing problem (BDP). Thus, the intention is to find certain permutation on both the layers of the bipartite graph such that the crossing count is minimized. Moreover, the restriction that the vertices in both the layers must have the same cardinality as imposed in [1] is also compromised. In our experiments, we consider two well-known crossing minimization heuristics i.e. MinSort and BaryCenter and the solution proposed in [1] as the basis of comparisons. In addition, the relationship between M-Sort and BC is also elaborated and their difference is noted for graphs with different densities. Results are compared upon input of variant density graphs.

The rest of the paper is organized as follows: In Section 2, Genetic Algorithm is discussed. The proposed solution is explained in Section 3. Experimental results and analysis are shown in Section 4. Finally, Section 5 concludes the paper.

2 Genetic Algorithm

Genetic Algorithm (GA) is a computational model of intelligence inspired by evolution introduced by John Holland in 1975 [6] [7]. The algorithm encodes a potential solution to a specific problem on a simple chromosome-like data structure and applies recombination operators to these structures [6]. The problems are not solved by reasoning logically about them, rather the populations of competing candidate solutions are spawned and then evolved to become better solutions through a process patterned after biological evolution [7]. Thus, increasingly powerful solutions emerge in a

Darwinian universe. Learning is viewed as a competition among a population of evolving candidate problem solutions [7].

3 Proposed Techniques

Two GA based techniques i.e., GA1 and GA2 are proposed. The main difference between these two techniques is the initialization step in the algorithm. The details are explained as follows:

3.1 Chromosome Representation

The implementation of a GA begins with a population of (typically random) chromosomes [6]. Chromosome is a string of genes and represents an individual i.e. a possible solution to a problem. A Gene is a basic unit which represents one characteristic of the individual. The value of each gene is called an allele. Each chromosome represents a point in the search space. A population is a collection of chromosomes. An appropriate chromosome representation is important for efficiency and reducing the complexity of the GA [7]. A chromosome for this problem is composed of permutations of two layers of bipartite graph as shown in Fig. 1.

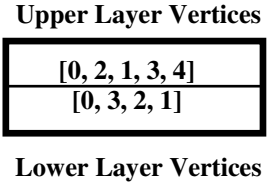


Fig. 1. Chromosome representation

3.2 Initialization

Based on different initialization strategies, two techniques are presented i.e., GA1 and GA2. In GA1, during initialization, the chromosomes in the population are created randomly i.e. both layers' permutations of bipartite graph are randomly enumerated; thus the initialization does not use any intelligence. The GA2 is a meta-heuristic in the sense that it uses M-Sort and BC during initialization phase. Instead of creating all the chromosomes randomly, the state of the bipartite graph in different iterations of M-Sort and BC is used to initialize/create a few chromosomes intelligently. The state means the permutations of vertices labels in both the layers where connectivity does not change. It takes usually several iterations to minimize the crossings in a bipartite graph using M-Sort and BC. In each iteration of M-Sort and BC, the permutation of the layers would change and thus may be used to create a new chromosome. In order to understand this, let us discuss an example for creating chromosomes using M-Sort as shown in Fig. 2. In this example, we create three chromosomes to be used in GA2. The format of the input data is also shown. The bipartite graph contains three vertices in both the layers having labels 0, 1 and 2 as highlighted in the input data.

The remaining data not highlighted contains '0' and '1'; where '0' indicates no edge and '1' indicates an edge between two vertices. The steps for crossing minimization under M-Sort are shown in Fig. 2.

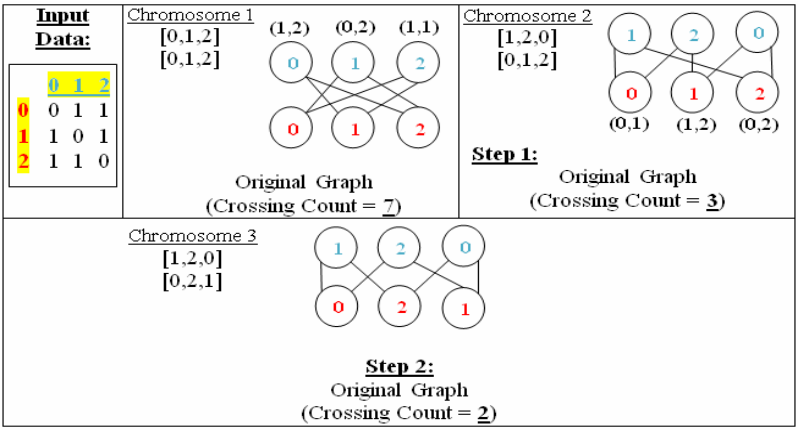


Fig. 2. Chromosome creation using MinSort

3.3 Stochastic Operator

GA evaluates the chromosomes and allocates reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to "reproduce" than those chromosomes which are poorer solutions [7]. For this, GA makes use of some stochastic operators mainly mutation and crossover. Both of these operators are used and the detail is given below:

Crossover: In our experiments, the X_Order1 method is used and the crossover rate is chosen equal to 1. In the X_Order1 method, the offspring inherits the elements between the two crossover points, inclusive, from the selected parent in the same order and position as they appear in that parent. The remaining elements are inherited from the alternate parent in the same order as they appear in that parent, beginning with the first position following the second crossover point and skipping over all elements already present in the offspring. In our approach, the chromosome is composed of two layer permutations of the bipartite graph thus X_Order1 crossover method is first applied on the upper layer and then on the lower layer between two selected parent chromosomes.

Mutation: The probability of mutation operator is kept constant to 0.03 which is found better experimentally. A swap method is used for mutation. In this method, two genes positions e.g. 'i' & 'j' from the parent are selected randomly and then swapped to form a new offspring.

Selection: The selection policy of individuals for the next generation is based on the survival of the fittest. New offspring are created equal to the size of the population. Population and offspring are sorted out on the basis of their relative fitness and best

individuals are selected to form the new population. For significant improvements in the results, duplicate chromosomes in the population are disallowed which definitely increases the computational effort but it encourages the new areas of the search space to be explored. As discussed, a chromosome is a search point in the search space and thus disallowing the duplicate search points means new search points to be examined or explored.

3.4 Fitness Function

Fitness function determines how much an individual is valuable to the problem at hand. In our case, crossing count is the basic criterion that can be used for indicating the value of an individual/chromosome. So, in order to evaluate the individual solution, the crossing count is used as a fitness function value, therefore, higher the fitness value, low valuable is the individual solution. Crossing count algorithms for bipartite graphs are very slow w.r.t. their time complexity. So, this would deteriorate the performance of GA due to multiple crossing counts of individuals in a population for multiple generations. Therefore, as a remedy, a fast $O(|E|.\log|V_{small}|)$ algorithm presented in [5] is used which overcomes this deterioration effect.

4 Experimental Results

The proposed techniques i.e., GA1 and GA2 are implemented in Visual Studio 2008 C# and run on Intel P-IV processor with 1 GB of RAM. The comparison among different techniques is done on the basis of various parameters. Each subsection below compares the techniques with respect to these parameters.

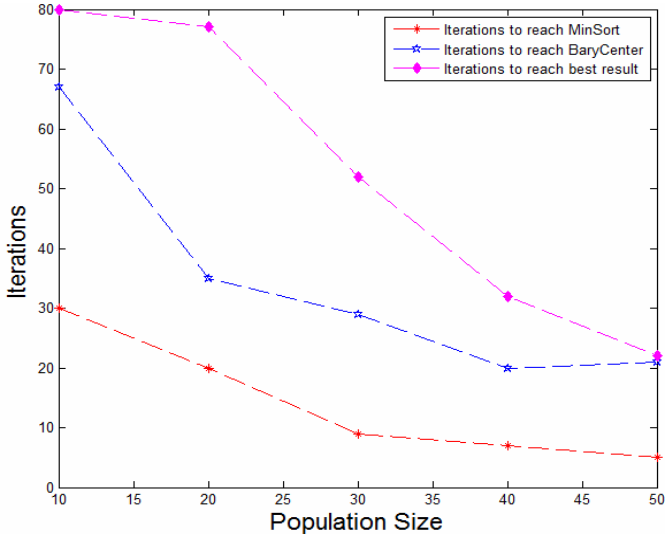


Fig. 3. GA vs Minsort and BC based on population size

4.1 Population Size

A random graph is generated with the size of 15 vertices on both the layers and the density is kept 20% as used in [1]. GA1 is run 50 times over the test graph and the average fitness gain is used for comparison with M-Sort and BC heuristics. Note that in Fig. 3, one curve depicts the GA's best result and the second and the third curves depict the number of iterations required to reach the results of M-Sort and BC heuristics, respectively. The result of GA1 is considered best when it does not improve in next twenty consecutive generations. Another interesting finding is the relationship between M-Sort and BC heuristics. It is found that BC outperforms the M-Sort heuristic with respect to minimizing the number of crossings, especially for high density graphs but it takes more iterations as compared to M-Sort.

4.1.1 GA1 vs. Minsort and BaryCenter

Fig. 3 shows the results of GA1 in comparison to MinSort and BC and the points in the graph are sprinkled based on different population sizes. On y-axis the numbers of iterations required are presented. For instance, our technique takes 30 iterations to reach MinSort results when population size is set to 10 and we get 107 crossings count as the best result after 80 iterations. We find no improvement in next twenty consecutive iterations after the completion of 80 iterations, thus consider best result of GA1 at this point. The working nature of the genetic algorithm is apparent that if population size is increased then the iterations required to acquire certain result are decreased. The population size '40' is found best during experiments as after that we find no significant improvement in the results, so considering it a saturation point in terms of performance gain. In general, we find that it requires large number of generations using GA1 to reach the results of BaryCenter as compared to the MinSort.

4.2 Graph Density

The known heuristics for LPP have the property of making larger relative error with low density graphs than with high density graphs [1]. We, therefore, test the effect of graph density in order to compare our technique with M-Sort and BC heuristics. For this purpose, we use random graphs of size 15, population size of 30 and mutation rate of 0.03. Graph size 15 means that there are 15 vertices in the upper layer as well as in the lower layer. This is just to compare our results with the one presented in [1], otherwise, this restriction is not relevant as evident from the representation of the chromosome. Moreover, even though the approach presented in [1] is based on GA and is used for crossing minimization, the solution presented in [1] is for a level permutation problem (LPP), whereas our proposed approach is a solution for the bipartite drawing problem (BDP). Therefore, it may be considered unfair to compare our approach with the one presented in [1], however, these comparisons are performed in order to show that rearranging both the layers of vertices of bipartite graph performs better (for crossing minimization) than just considering a single layer for possible rearrangements.

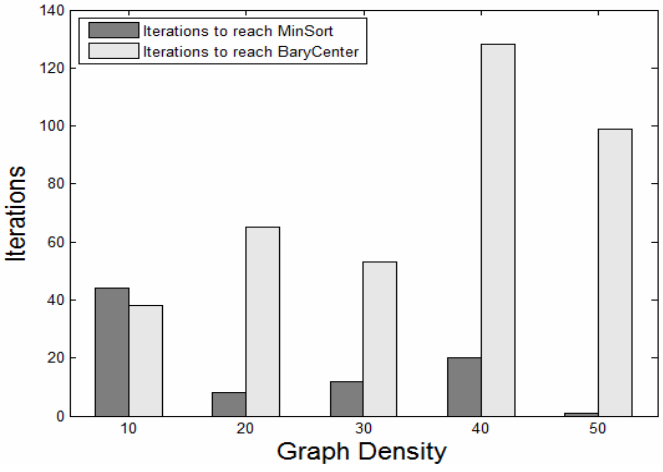


Fig. 4. Iterations to acquire the results of MinSort and BC with variant graph densities

The GA1 results for this experiment are calculated as the average of 50 runs for different values of graph densities as shown in Fig. 4 for M-Sort and BC. Graph density represents the number of edges in the graph. For example, if we have total (n=30) vertices (15 vertices in each layer) in the bipartite graph and the graph density is (d=20%) then there will be 45 edges ((n*n)*d/ 100) in total.

4.2.1 GA1 vs. Minsort and BaryCenter

Fig. 4 shows the graphical representation of the results obtained for the comparison of GA1 with MinSort and BaryCenter based on different graph densities. We can see from the results in Fig. 4 that we need small number of iterations of GA1 to reach the MinSort results for highly dense graphs as we have observed that MinSort makes relatively large error in case of highly dense graphs. BaryCenter has the property of making larger relative error with low-density graphs than with high-density graphs [1]. So, based on this observation, we can see in Fig. 4 that GA1 needs large number of iterations to reach BaryCenter results in case of highly dense graphs.

4.3 Crossing Minimization

We compare GA1 with MinSort and BaryCenter such that we allow a total of 500 generations for GA1 and analyze possible average crossing minimizations using GA1. We generate a random bipartite graph with the size of 15 vertices on both the layers and density is kept 20%. We calculate the average crossing count after running the GA1 50 times with the mutation rate of 0.03 and the population size of 40. From experiments, we show that GA1 performs much better than M-Sort and B-Center with average crossing count of 77 as compared to M-Sort (166) & B-Center (123) where the original crossing count are 460.

4.4 Comparison of GA1 with GA2

We now present the comparison of GA1 and GA2 based on different graph densities and different sizes of bipartite graphs generated randomly. The comparison based on different graph densities is shown in Table 1. The constant parameters during these experiments are given as: (Graph Size = 15*15, Generations = 200, Mutation rate = 0.03, Crossover rate = 1 and Total Runs = 50).

The comparison of GA1 and GA2 for different graph sizes generated randomly is shown in Table 2. The constant parameters during these experiments are given as: (Graph Density = 30%, Population Size = 40, Generations = 500, Mutation rate = 0.03, Crossover rate = 1 and Total Runs = 50). It is clear from Table 1 and Table 2 that GA2 is able to minimize the crossings in the bipartite graph much better than GA1 since GA2 uses an intelligent initialization scheme as discussed in Section 3.2.

Table 1. Comparison of GA1 and GA2 based on graph densities and population size (O.C = Original Crossings and A.C.C = Average Crossing Count)

Density = > / Population	10% (O.C = 123)		20% (O.C = 460)		30% (O.C = 918)	
	GA2 A.C.C	GA1 A.C.C	GA2 A.C.C	GA1 A.C.C	GA2 A.C.C	GA1 A.C.C
10	10	16	110	159	359	363
20	4	15	106	148	352	362
30	3	10	105	125	329	352
40	3	9	103	120	327	339
50	3	6	95	121	321	323

Table 2. GA1 vs GA2 (different graph Sizes) (A.C.C = Average Crossing Count)

Graph Size	GA2 A.C.C	GA1 A.C.C	Original Crossing Count
20*20	1173	1346	3634
30*30	8291	8309	16697
40*40	26040	26081	55557
50*50	85824	86089	136542
100*100	905916	905966	2197974

5 Conclusion

In this paper, we proposed two techniques i.e., GA1 and GA2 based on genetic algorithm for the bipartite drawing problem (BDP). The proposed techniques are also compared with previously known heuristics such as MinSort and BaryCenter as well as a genetic algorithm based level permutation problem (LPP). We have found that on average, the proposed techniques always outperform BaryCenter on low-density graphs and MinSort on high-density graphs. The behavior shown during experiments by the genetic algorithm in our techniques is quite good with respect to average number of crossing minimization. There is, of course, a saturation point especially in evolutionary computing based techniques such as GA where the improvement in the results stops or improves very slowly. The parameters we used in our experiments for

GA1 and GA2 are; population size of 10-50, elitism is at least one individual, cross-over rate is 1 and mutation rate is 0.03.

References

1. Mäkinen, E., Sieranta, M.: Genetic Algorithms for Drawing Bipartite Graphs. International Journal of Computer Mathematics 53(3&4), 157–166 (1994)
2. Bienstock, D.: Some provably hard crossing number problems. In: Proc. 8th Annual ACM Symposium on Computational Geometry, pp. 253–260 (1990)
3. Garey, M.R., Johnson, D.S.: Crossing number is NP-complete. SIAM J. Algebraic Discrete Methods 4, 312–316 (1983)
4. Tutte, W.T.: Toward a Theory of Crossing Numbers. J. Comb. Theory 8(1), 45–53 (1970)
5. Barth, W., Mutzel, P., Junger, M.: Simple and Efficient Bilayer Cross Counting. Journal of Graph Algorithms and Applications (JGAA) 8(2), 179–194 (2004)
6. Whitley D.: A Genetic Algorithm Tutorial, `http://www.cs.uga.edu/~potter/CompIntell/ga_tutorial.pdf`
7. Engelbrecht, A.P.: Computational Intelligence: An Introduction. J. Wiley & Sons, Chichester (2007)
8. Watkins, M.E.: A special crossing number for bipartite graphs: a research problem. Ann. New York Acad. Sci. 175, 405–410 (1970)
9. Battista, G.D., Eades, P., Tamassia, R., Tollis, I.: Algorithms for drawing graphs: an annotated bibliography. Comp. Geometry: Theory and Applications 4, 235–282 (1994)
10. Koebe, M., Knöchel, J.: On the block alignment problem. J. Inf. Process. Cybern. EIK 26, 377–387 (1990)
11. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Trans. Syst. Man Cybern. SMC-11, 109–125 (1981)
12. Abdullah, A.: "Data Mining Using the Crossing Minimization Paradigm," –Ph.D. Thesis, University of Stirling (2007)
13. Catarci, T.: The Assignment Heuristic for Crossing Reduction. IEEE Transactions on Systems, Man and Cybernetics 21, 515–521 (1995)
14. Camel, D., Irene, F.: Breaking Cycles for Minimizing Crossings. Journal of Experimental Algorithmics (JEA) 6 (2001)
15. Stallmann, M., Brglez, F., Ghosh, D.: Heuristics, Experimental Subjects, and Treatment Evaluation in Bigraph Crossing Minimization. J. Exp. Algorithmics 6 (2001)
16. Zheng, L., Song, L., Eades, P.: Crossing Minimization Problems of Drawing Bipartite Graphs in Two Clusters. In: ACM Asia-Pacific Symposium on Information Visualization, vol. 109, pp. 33–37 (2005)
17. Ezziane, Z.: Experimental Comparison between Evolutionary Algorithm and Barycenter Heuristic for the Bipartite Drawing Problem. J. Computer Science 3(9), 717–722 (2007)
18. Laguna, M., Marti, R., Vails, V.: Arc Crossing Minimization in Hierarchical Digraphs with Tabu Search. Computers and Op. Research 24(12), 1175–1186 (1997)
19. Martí, R.: Arc Crossing Minimization in Graphs with GRASP. IIE Transactions 33, 913–919 (2004)

Evaluation of Two-Stage Ensemble Evolutionary Algorithm for Numerical Optimization

Yu Wang, Bin Li, Kaibo Zhang, and Zhen He

Natural Inspired Computation and Application Laboratory (NICAL)
Department of Electronic Science and Technology
University of Science and Technology of China (USTC)
Tel.: +86-551-3601802
wyustc@mail.ustc.edu.cn

Abstract. In many challenging numerical optimization problems, the conflict between exploitation and exploration abilities of EAs must be balanced in an effective and efficient way. In the previous research, in order to address this issue, the Two-Stage ensemble Evolutionary Algorithm (TSEA) was originally proposed for engineering application. In TSEA, the optimization is divided into two relatively separate stages, which aims at handling the exploitation and exploration in a more reasonable way. In this paper, we try to extend the application area of TSEA from specific engineering problems to general numerical optimization problems by altering its sub-optimizers. The experimental studies presented in this paper contain three aspects: (1) The benefits of the TSEA framework are experimentally investigated by comparing TSEA with its sub-optimizers on 26 test functions; then (2) TSEA is compared with diverse state-of-the-art evolutionary algorithms (EAs) to comprehensively show its advantages; (3) To benchmark the performance of TSEA further, we compare it with 4 classical memetic algorithms (MAs) on CEC05 test functions. The experimental results definitely demonstrate the excellent effectiveness, efficiency and reliability of TSEA.

1 Introduction

Numerical optimization problems widely exist in scientific and engineering applications [1]. Since the competition of searching for optimal working conditions in engineering application has become much more furious, the development of the optimization methods always lead to rapid improvement of the competitive strength [2,3,4]. However, these problems always have different characteristics and rise more and more challenges to the optimization methods [5,6]. Among these challenges, the balance between exploitation and exploration is an important one that has attracted public attention.

In the previous research, many attempts have been carried out to synthesize diverse kinds of merits from different optimization techniques, which surely strengthens the universality in one algorithmic framework. Among these attempts, memetic algorithm (MA) is a typical example [7,8,9]. The major idea of MAs is to make up the demerit of less exploration ability of global search algorithms by embedding efficient local search techniques, which can more easily obtain a near optimal solution. Another important way is to combine numbers of new offspring generating strategies from one or multiple evolutionary algorithms (EAs) in a parallel way [4,10,11,12,13,14].

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 539–548, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Different from the above works, an Estimation of Distribution and Differential Evolution cooperation (ED-DE), which adopt a serial algorithmic framework to extract the merits from both estimation of distribution and differential evolution, is originally proposed in [2]. At each iteration, ED-DE only utilizes self-adaptive mixed distribution based univariate estimation of distribution algorithm (MUEDA) [6] or modified differential evolution (MDE). In ED-DE, the whole optimization procedure is divided into two relatively separate stages, which mainly focus on exploitation and exploration respectively. The essence of ED-DE is to completely solve the simple problems in the first stage, and obtain a good initial status for the second stage when the problems are too hard to solve by only implementation of MUEDA. Following the similar line of thinking, Wang and Li [5] extend the two-stage search idea to large scale global optimization.

It can be observed that the two-stage search idea has just been applied to specific optimization problems, while inevitably ignores some requirements of general usage, such as rotation. In this paper, we try to extend the application area of TSEA from specific engineering problems to general numerical optimization problems by altering its sub-optimizers. The sub-optimizers used in TSEA are the Covariance Matrix Adaptation Evolution Strategy (CMAES) [15] and Self-adaptive Learning based Particle Swarm Optimization (SLPSO) [4]. The reasons of implementing these two algorithms can be summarized as follows:

- (1) CMAES shows particularly reliable and excellent performance for local optimization. In the Matlab source code of CMA-ES, it is stated that its convergence speed is ten times slower than the mathematical local search Davidon, Fletcher and Powell Strategy (DFP) [16], but its robustness for difficult problems is far more excellent. Given an initial individual, CMA-ES performs iterative self-adaptive Gaussian-based mutation and recombination. Especially, for the rotated problems, CMA-ES can effectively handle them by adapting a covariant matrix during the optimization procedure [17,18].
- (2) SLPSO simultaneously applies four PSO based search strategies, whose associated probabilities of using different strategies are self-adaptively learnt by considering the ability of generating better quality solutions in the past generations. In this case, the robustness and universality of SLPSO can be highly improved. Especially, the effectiveness of SLPSO, which is the main requirement of the second optimization stage in TSEA, has been experimentally verified in [4].

In order to fully show the advantages of TSEA, we design three experimental studies: (1) The benefits of the TSEA framework are experimentally investigated by comparing TSEA with its sub-optimizers on 26 test functions; then (2) TSEA is compared with diverse state-of-the-art evolutionary algorithms (EAs) to comprehensively show its advantages; (3) To benchmark the performance of TSEA further, we compare it with 4 classical memetic algorithms (MAs) on CEC05 test functions.

The remainder of this paper is structured as follows: In the next section, the TSEA framework is depicted. Experimental comparison between TSEA and its sub-optimizers is shown and discussed in the Section III and in Section IV, further experimental studies are presented. Finally, the contributions of this paper are summarized and the future work is outlined in Section V.

Table 1. Procedure of TSEA

TSEA
Input: – Optimization task ; – a transformation criterion (trigger) (it depends on different algorithms used in the first stage); – a termination condition; **Output:** The best solution found.
Step 0) Initialization: Randomly initialize the population X_t. Set $t = 0$.
Step 1) The first optimization stage: Perform new offspring generating strategy 1 for one iteration to update X_t. Set $t = t + 1$.
Step 2) Trigger Determine whether to trigger the second optimization stage or continue the first optimization stage. In case of former, go to step 3), otherwise, go back to step 2).
Step 3) The second optimization stage: Perform iterative new offspring generating strategy 2.
Step 4) Terminate and output.

2 Algorithm

As introduced in Section I, TSEA combines two EAs to form a new serial cooperative optimizer. More promising results are expected when the characteristic merits of both EAs are successfully united in the cooperative optimizer. Generally speaking, the motivation of combining two perspective EAs in a successive manner are:

– When a new black-box problem is given, its properties are completely unknown. Therefore, the choice of the optimization maybe sightless for users. To address this issue, TSEA adopts an efficient algorithm to accomplish the pre-trial. If the problem is simple, TSEA can completely solve it in the first stage. Contrary, when the problem is too hard, e.g. the problem contains too many local optima, the first stage can also achieve a good initial status for the second stage within a limited computational cost.
– The situation that the problem cannot be solved in the first stage implies that it is a hard task. The second stage, which implements an effective algorithm, is launched with the initial status inherited from the first stage. The effective algorithm, which mainly focuses on exploration, always consumes larger computational cost in accomplishing one search procedure. If we directly use this algorithm in the first stage, much computational cost may be wasted for the easy problems.

The detailed procedure of the TSEA is illustrated in Table 1, where the new offspring generating strategies 1 and 2 are CMAES and SLPSO respectively.

3 Experiment Comparison between TSEA and Its Sub-optimizers

In order to demonstrate the benefits of TSEA framework over its sub-optimizers, we perform TSEA versions and its sub-optimizers on the test suit shown in Table 2. The first two groups contain separable problems. Test functions of group 2 have mis-scaled variables or noisy landscapes, which are much more challenging for optimization. For group 3 problems, the classical test functions are rotated by $z = M(x - o)$, where M

Table 2. Classical benchmark problems to be minimized

Group 1: Classical test functions (non-linear, separable, scalable) ($z = x - o$)				
Num	S	R	Problems	Objective function
fun1	√		**Shifted Sphere**	$f_1(x) = \sum_{i=1}^{n} z_i^2 + f_{bias}$
fun2	√		**Shifted Schwefel 1.2**	$f_2(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} z_j^2) + f_{bias}$
fun3	√		**Shifted Schwefel 2.22**	$f_3(x) = \sum_{i=1}^{n} \|z_i\| + \prod_{i=1}^{n} \|z_i\| + f_{bias}$
fun4	√		**Shifted Schwefel 2.21**	$f_4(x) = \max \|z_i\| + f_{bias}$
fun5	√		**Shifted Rochenbrock**	$f_5(x) = \sum_{i=1}^{D} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$
fun6	√		**Shifted Ackley**	$f_6(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}) + e - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)) + 20 + f_{bias}$
fun7	√		**Shifted Griewank**	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} cos(\frac{z_i}{\sqrt{i}}) + f_{bias}$
fun8	√		**Shifted Rastrigin**	$f_8(x) = \sum_{i=1}^{n} (z_i^2) - 10cos(2\pi z_i) + 10 + f_{bias}$
fun9	√		**Noncontinues Rastrigin**	$f_9(x) = \sum_{i=1}^{n} (y_i^2) - 10cos(2\pi y_i) + 10, y_{i+1} = \begin{cases} z_i, if \ \|z_i\| < 1/2 \\ round(2 * z_i)/2, otherwise, \end{cases}$
fun10	√		**Shifted Penalized 1**	$f_{10} = \frac{\pi}{30}\{10\sin^2 + \sum_{i=1}^{2} 9(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1} + (y_n - 1)^2)] + \sum_{i=1}^{30} u(x_i, 10, 100, 4)\}$, u and y is shown in Appendix
fun11	√		**Shifted Penalized 2**	$f_{11} = 0.1\{\sin^2(\pi 3x_1) + \sum_{i=1}^{2} 9(x_i - 1)^2 \cdot [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2\} \cdot [1 + \sin^2(2\pi x_3 0)] + \sum_{i=1}^{30} u(x_i, 5, 100, 4)\}$, u and y is shown in Appendix
Group 2: Test functions that are mis-scaled or with noise ($z = x - o$)				
fun12	√		**Shifted Rochenbrock100**	$f_{12}(x) = \sum_{i=1}^{D} (100((a_i z_i)^2 - (a_{i+1} z_{i+1}))^2 + ((a_i z_i) - 1)^2) + f_{bias}, a_i = 10^{\frac{i-1}{D-1}}$
fun13	√		**Shifted Rastrigin10**	$f_{13}(x) = \sum_{i=1}^{n} ((a_i z_i)^2) - 10cos(2\pi(a_i z_i)) + 10 + f_{bias}, a_i = 10^{\frac{i-1}{D-1}}$
fun14	√		**Shifted Rastrigin1000**	$f_{14}(x) = \sum_{i=1}^{n} ((a_i z_i)^2) - 10cos(2\pi(a_i z_i)) + 10 + f_{bias}, a_i = 1000^{\frac{i-1}{D-1}}$
fun15	√		**Noise Schwefel 1.2**	$f_{15}(x) = (\sum_{i=1}^{n} (\sum_{j=1}^{i} z_j^2))(1 + 0.4\|N(0,1)\|) + f_{bias}$
Group 3: Rotated test functions (non-linear, non-separable, scalable) ($z = M(x - o)$)				
Num	S	R	Problems	Objective function
fun16	√	√	**Rotated Sphere**	$f_{16}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}$
fun17	√	√	**Rotated Tablet**	$f_{17}(x) = (1000x_1)^2 + \sum_{i=2}^{D} z_i^2 + f_{bias}$
fun18	√	√	**Rotated Ellipse**	$f_{18}(x) = \sum_{i=1}^{D} (20^{\frac{i-1}{D-1}} z_i)^2 + f_{bias}$
fun19	√	√	**Rotated diff pow**	$f_{19}(x) = \sum_{i=1}^{D} z_i^{2+a_i} + f_{bias}, a_i = 10^{\frac{i-1}{D-1}}$
fun20	√	√	**Rotated Schwefel 2.21**	$f_{20}(x) = \max \|z_i\| + f_{bias}$
fun21	√	√	**Rotated Rochenbrock**	$f_{21}(x) = \sum_{i=1}^{D} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}$
fun22	√	√	**Rotated Ackley**	$f_{22}(x) = -20 \cdot exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}) + e - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi z_i)) + 20 + f_{bias}$
fun23	√	√	**Rotated Griewank**	$f_{23}(x) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} cos(\frac{z_i}{\sqrt{i}}) + f_{bias}$
fun24	√	√	**Rotated Rastrigin**	$f_{24}(x) = \sum_{i=1}^{n} (z_i^2) - 10cos(2\pi z_i) + 10 + f_{bias}$
fun25	√	√	**Noise Schwefel 1.2**	$f_{25}(x) = (\sum_{i=1}^{n} (\sum_{j=1}^{i} z_j^2))(1 + 0.4\|N(0,1)\|) + f_{bias}$
fun26	√	√	**Noise Quadric**	$f_{26}(x) = \sum_{i=1}^{n} iz_i^4 + random[0,1) + f_{bias}$

is an orthogonal rotation matrix, to avoid local optima lying along the coordinate axes while retaining the properties of the test functions.

Before presenting the experiment, we first define the comparison criterion: the ranking of the algorithms is based on the success performance SP [17], which is defined as follows:

$$SP = mean(\text{function evaluations of successful runs}) \times \frac{\text{all runs}}{\text{number of successful runs}} \tag{1}$$

Small values of SP are preferable. As shown in [4,11,14,17], the empirical cumulative distribution of normalized SP can strongly benchmark the performance of multiple algorithms.

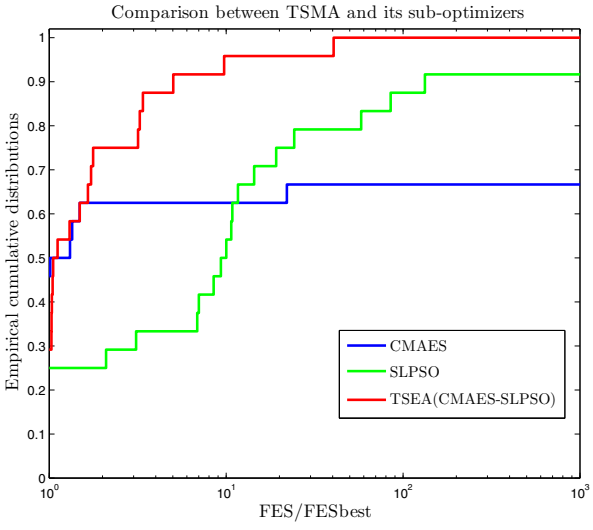


Fig. 1. Empirical distribution of normalized success performance of TSEA and its sub-optimizers

In this experiment, we set the dimension size of all test functions to be 30 and the maximum number of function evaluations to be 300000. For all test functions, we perform 30 independent runs with each algorithm. A run is considered to be successful if at least one solution was discovered during its course whose fitness value is not worse than $(fit(x^*) + 1e - 5)$.

It is observed from Fig. 1 that CMA-ES provides top convergence speed for almost 50% problems. However, for over 30% problems, it fails completely, which implies that it is not sufficient to implement only once for reliably solving diverse tasks. This viewpoint is the main reason of applying restart strategy in [17,18]. SLPSO can solve more problems than CMAES, however, its convergence speed is relatively lower. With the good cooperation of the both techniques, the overall performance curves of TSEA is the highest, which means TSEA has the best universality and excellent efficiency.

4 Experiment Comparison between TSEA and State-of-the-Art EAs and MAs

4.1 Experiment Comparison with State-of-the-Art EAs

In order to show the superiority of TSEA over the effective global search methods, we compared it with eight recently proposed state-of-the-art EAs:

- SaDE: self-adaptive differential evolution [14];
- jDE: differential evolution with parameter adaption [19].
- MUEDA: self-adaptive mixed distribution based univariate EDA [6];
- CLPSO: comprehensive learning particle swarm optimizer [20];
- FIPS-PSO: fully informed PSO [21];

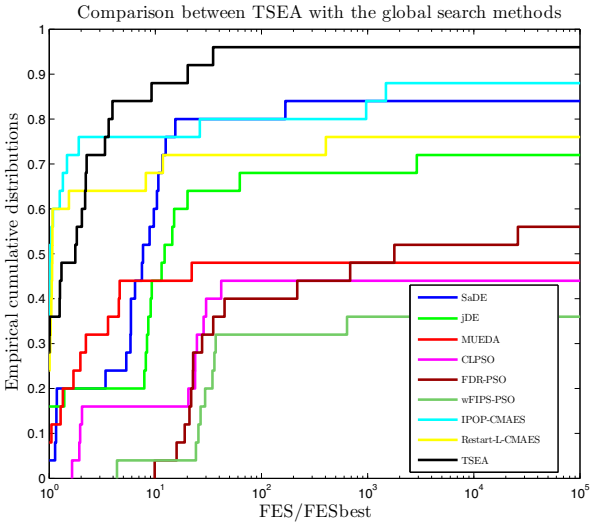


Fig. 2. Comparison between TSEA and state-of-the-art EAs

- FDR-PSO: Fitness-distance-ratio based PSO [22];
- IPOP-CMAES: global CMAES variant with increasing population size and restart strategy [18];
- Restart-L-CMAES: local CMAES variant with restart strategy [17];

Especially, the self-adaptively utilize offspring generating strategies to strengthen the robustness. CLPSO has exhibited the top capability of handling multi-modal problems [20] among the PSO variants. IPOP-CMAES showed top performance in function optimization competition of IEEE Congress of Evolutionary Computation 2005 (CEC05). The essential of IPOP-CMAES and Restart-L-CMAES is to restart CMA-ES version when the search fails. Therefore, their searches are made of multiple runs of CMA-ES on many problems.

Fig. 2 depicts the comparison between TSEA and eight state-of-the-art EAs. The superior capacity in universality of SaDE compared the other DE variants has been experimentally verified in [14]. It is observed that SaDE significantly outperform jDE, MUEDA and the other PSOs. Among all the algorithms, only IPOP-CMAES and Restart-L-CMAES use the restart strategy, which can highly improve their performance. However, when the problems contain too many local optima, it is still too hard to locate the initial individuals to an area near the global optimum for them. Therefore, they just can provide a comparable overall result compared with SaDE without restart strategy. It is interesting to see that TSEA has significantly better convergence speed and universality in Fig. 1. For almost 30% problems, TSEA provides the fastest optimization speed. Furthermore, when its empirical cumulative distribution curve reaches the top, the normalized SP value is still small. In Fig. 3, the comparison of optimization curves of algorithms without restart strategy on four representative problems again confirms

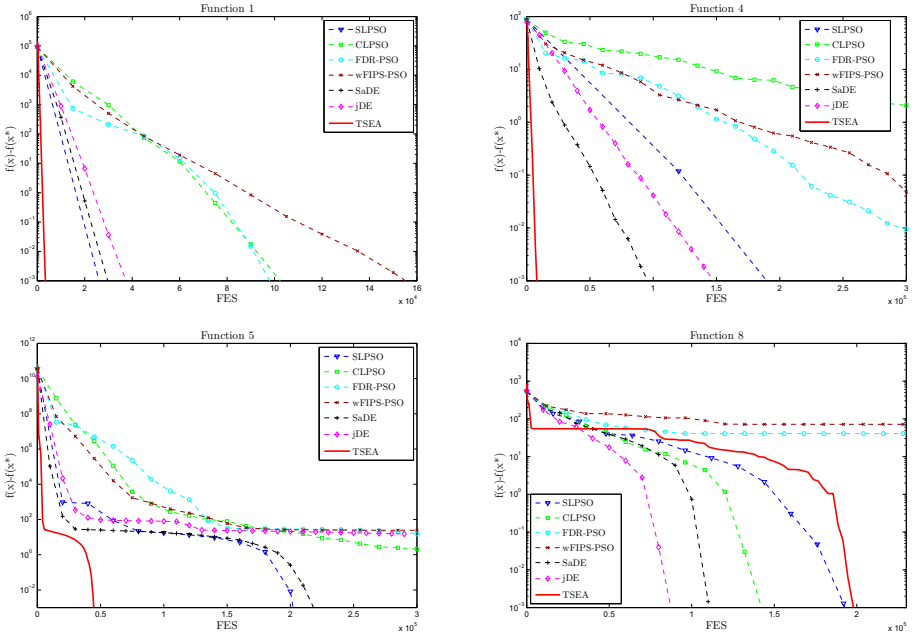


Fig. 3. Optimization curves of function optimization with 30 D. (Sphere, Schwefel 2.21, Rochenbrock and Rastrigin).

this viewpoint. Therefore, the advantages of the TSEA compared with effective EAs are definitely verified.

4.2 Experiment Comparison with State-of-the-Art MAs

To demonstrate the advantages of TSMA over the other MAs, we compare TSMA with three MA versions from CEC05 function optimization competition and another recently proposed MA for general problems:

- SaDE-L: self-adaptive differential evolution with local search DFP [23];
- DMS-L-PSO: dynamic multi-swarm particle swarm optimizer with local search [25];
- BLX-MA: real-coded memetic algorithm with adaptive local-search probability and local search length [26].
- APrMA: adaptive probabilistic memetic algorithm [8];

In general, they have different strengthes. APrMA is specially proposed for general problems, and has shown distinct progress compared with diverse MAs.

In this experiment, a run is considered to be successful if at least one solution was discovered during its course whose fitness value is not worse than $(fit(x^*) + 1e - 6)$ for the functions 1 to 5 and $(fit(x^*) + 1e - 2)$ for the other functions. The overall performance on the first 16 problems in empirical accumulative distribution of normalized SP is described in Fig. 4.

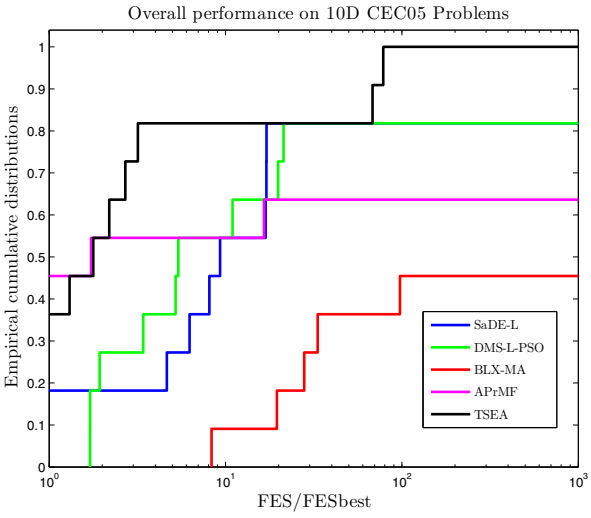


Fig. 4. Empirical distribution of normalized success performance on CEC05 function optimization tasks

From Fig. 4, we can observe that TSEA is definitely the best, because of the higher optimization speed and the higher number of solved problems. For the other algorithms, the performance of APrMA, SaDE-L and DMS-L-PSO is similar and cannot solve no more than 80% of the problems that can be solved by TSEA.

5 Conclusion

In the previous research, the idea of two-stage search has shown promising performance on specific engineering applications. This paper extends its application to general numerical optimization. The experimental results definitely verify the superior performance of TSEA over its sub-optimizers, state-of-the-art EAs and MAs. In the recent future, we expect to apply it to solve more engineering optimization problems.

Acknowledgment

The authors would like to thank Prof. P. N. Suganthan, Mr. Nikolaus Hansen and Prof. Y. S. Ong for providing the codes of their research group.

This work was partially supported by the National Natural Science Foundation of China (No. 61071024, U0835002), the Chinese Academy of Science (CAS) Special Grant for Postgraduate Research, Innovation and Practice, and Graduate Innovation Fund of University of Science and Technology of China, the USTC Innovation Fund for Young Researchers.

References

1. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. 3(2), 82–102 (1999)
2. Wang, Y., Li, B., Weise, T.: Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. Information Sciences 180, 2405–2420 (2010)
3. Wang, Y., Li, B., Chen, Y.B.: Digital IIR filter design using multi-objective optimization evolutionary algorithm. Appl. Soft Comput. J. (2010), doi:10.1016/j.asoc, 05.034
4. Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., Tian, Q.: Self-Adaptive Learning based Particle Swarm Optimization. Information Sciences (accepted in press)
5. Wang, Y., Li, B.: Two-stage based Ensemble Optimization for Large-Scale Global Optimization. In: Proc. the IEEE Congress on Evolutionary Computation (CEC 2010), Barcelona, Spain, pp. 4488–4495 (July 2010)
6. Wang, Y., Li, B.: A Self-adaptive Mixed Distribution Based Uni-variate Estimation of Distribution Algorithm for Large Scale Global Optimization. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimization, Hardcover. Studies in Computational Intelligence, vol. 193, pp. 171–198. Springer, Heidelberg ISBN: 978-3-642-00266-3
7. Ong, Y.S., Keane, A.J.: Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. Evol. Comput. 8(2), 99–110 (2004)
8. Nguyen, Q.H., Ong, Y.-S., Lim, M.H.: A Probabilistic Memetic Framework. IEEE Trans. Evol. Comput. 13(3), 604–623 (2009)
9. Ong, Y.S., Lim, M.H., Chen, X.S.: Research Frontier: Memetic Computation - Past, Present & Future. IEEE Computational Intelligence Magazine 5(2), 24–36 (2010)
10. Sun, J., Zhang, Q.F., Tsang, E.: DE/EDA: A new evolutionary algorithm for global optimization. Information Sciences 169, 249–262 (2005)
11. Vrugt, J.A., Robinson, B.A., Hyman, J.M.: Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces. IEEE Trans. Evol. Comput. 13(2), 243–259 (2009)
12. Peña, J.M., Robles, V., Larrañaga, P., Herves, V., Rosales, F., Pérez, M.S.: GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 361–371. Springer, Heidelberg (2004)
13. Peña, J.M., Robles, V., Larrañaga, P., Herves, V., Rosales, F., Pérez, M.S.: GA-EDA: Hybrid evolutionary algorithm using genetic and estimation of distribution algorithms. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 361–371. Springer, Heidelberg (2004)
14. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. IEEE Trans. Evol. Comput. 13(2), 398–417 (2009)
15. Hansen, Ostermeier: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)
16. Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. Computer Journal 7(4), 303–307 (1964)
17. Auger, A., Hansen, N.: Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1777–1784 (2005)
18. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1769–1776 (2005)
19. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Trans. Evol. Comput. 10(6), 646–657 (2006)

20. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. IEEE Trans. Evol. Comput. 10(3), 281–295 (2006)
21. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: Simpler, maybe better. IEEE Trans. Evol. Comput. 8(3), 204–210 (2004)
22. Peram, T., Veeramachaneni, K., Mohan, C.K.: Fitness-distance-ratio based particle swarm optimization. In: Proc. Swarm Intelligence Symp., pp. 174–181 (2003)
23. Qin, A.K., Suganthan, P.N.: Self-adaptive Differential Evolution Algorithm for Numerical Optimization. In: Proc. IEEE Congress on Evolutionary Computation (CEC), pp. 1785–1791 (2005)
24. Rökkönen, J., Kukkonen, S., Price, K.V.: Real-parameter optimization with differential evolution. In: Proc. IEEE Congress on Evolutionary Computation (CEC), pp. 506– 513 (2005)
25. Liang, J.J., Suganthan, P.N.: Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search. In: Proc. IEEE Congress on Evolutionary Computation, pp. 522–528 (2005)
26. Molina, D., Herrera, F., Lozano, M.: Adaptive Local Search Parameters for Real-Coded Memetic Algorithms. In: Proc. IEEE Congress on Evolutionary Computation, pp. 888–895 (2005)

A Novel Genetic Programming Algorithm for Designing Morphological Image Analysis Method*

Jun Wang and Ying Tan**

Key Laboratory of Machine Perception (Ministry of Education), Peking University
Department of Machine Intelligence, School of EECS, Peking University, P.R. China
wangjun@cis.pku.edu.cn ytan@pku.edu.cn

Abstract. In this paper, we propose an applicable genetic programming approach to solve the problems of binary image analysis and gray scale image enhancement. Given a section of original image and the corresponding goal image, the proposed algorithm evolves for generations and produces a mathematic morphological operation sequence, and the result performed by which is close to the goal. When the operation sequence is applied to the whole image, the objective of image analysis is achieved. In this sequence, only basic morphological operations— erosion and dilation, and logical operations are used. The well-defined chromosome structure leads brings about more complex morphological operations can be composed in a short sequence. Because of a reasonable evolution strategy, the evolution effectiveness of this algorithm is guaranteed. Tested by the binary image features analysis, this algorithm runs faster and is more accurate and intelligible than previous works. In addition, when this algorithm is applied to infrared finger vein gray scale images to enhance the region of interest, more accurate features are extracted and the accuracy of discrimination is promoted.

1 Introduction

Digital images processing on computers have been applied to many fields like pattern recognition, robotic vision, biomedical image analysis, and biometrics, etc [10][12][16]. Recent years, a variety of evolutionary methods are used to solve the problem of discovering algorithm for image processing [2][7][11][13]. Most of them concentrate on that the image analysis problems can be re-framed as filtering problems, and use genetic algorithm (GA) and genetic programming (GP) to produce a set of standard filters [7][11][12]. However, all these filter-based GA(GP) methods need complicated formulations which require a large amount of analysis and computation [11].

Mathematic morphological is a powerful non-linear tool for extracting image components, which is useful in the representation and description of region shape, such as boundaries, skeletons, and the convex hulls [5]. Morphological operators aim at

* This work is supported by National Natural Science Foundation of China (NSFC), under grant number 60875080 and 60673020, and partly supported by the National High Technology Research and Development Program of China (863 Program), with grant number 2007AA01Z453.

** Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 549–558, 2011.
© Springer-Verlag Berlin Heidelberg 2011

extracting relevant structures of the image considered as a set through its sub graph representation, which is achieved by probing the image with another set of the known shapes called *structural element* (SE) [15]. Searching suitable morphological operation sequence and corresponding SEs in a big searching space is suited to the genetic programming [13]. Some researchers use an evolutionary morphological approach to analyze image. Yoda et al. explore the possibility of obtaining mathematic morphological algorithm for binary images by means of GA [18]. Harvey et al. describe a technique by GA for the optimization of multidimensional gray scale soft morphological filters [6]. They use this technique in the spatiotemporal domain for applications in automatic film restoration. Quintana et al. propose an approach of morphological binary image analysis based on GP [13][14]. Their algorithm is constructed by logic operators and the basic morphological operators — erosion and dilation — with a group of manually chosen SEs. This algorithm evolves to generate morphological operation sequence which converts a binary image into the target image contained just a particular feature of interest. Additional, they prove that it is possible to evolve good morphological methods by using GP. Ballerini et al. propose a GP method without a goal image and the morphological operations are not used for noise reduction or segmentation, but for image classification [1].

In this paper, we propose a applicable genetic programming algorithm to automatic generate methods for binary image analysis and gray scale image enhancement. Given a section of original image and the corresponding goal image, this algorithm automatically produces a mathematic morphological operation sequence, and the result by which is close to the goal. Afterwards, when the operation sequence is applied to the whole image, the objective of image analysis is achieved. This paper is organized as follows. Section 2 presents a novel genetic programming algorithm with mathematic morphological operations. The experimental results of binary images analysis and gray scale image enhancement are reported in Section 3. Finally, Section 4 gives the concluding remarks of this paper.

2 Proposed GP Algorithm

2.1 Definition of GP Algorithm

The genetic programming provides an approach to the problem of finding a computer program to solve a problem [9]. In our problem, by giving an original image and a learning target— goal image, the GP algorithm automatically produces a sequence of operations, which is applied to the original image and the obtained result is very close to the goal.

Definition of Chromosome. A chromosome in this algorithm is an individual, i.e., a method, which is composed by a sequence of genes, illustrated in Fig 1.

We use gene to express the primitive operation set. One single gene is a primitive unit of morphological operation accompanied with logical operations. The number of genes in each chromosome may be different while each gene has a fixed length. And two processing sequences are kept in one chromosome, which are connected by logical operations.

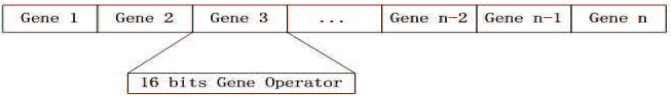


Fig. 1. Chromosome definition

2.2 Definition of Gene

The gene structure is shown in Table 1. Each gene has 16 bits and only 14 bits have real meanings since the first two bits are used for alignment. All these bits are classified into two parts: one part expresses logical operation and the control flow, and the other part expresses morphological operation.

Table 1. The meaning of each bit in a gene

	Index	Meaning
Unused	1	null
	2	null
logical and control	3	switch
	4	storage flag
	5	direct/difference flag
	6	logical operation flag
	7-8	logical operator
morphological operation	9	erosion/dilation flag
	10-11	SEs size
	12	SE class
	13-16	SEs index

Direct/Difference flag. The direct/difference flag means which one will be chosen as current output, the morphological operation result directly or the arithmetic difference between the result and the input. This operation brought forward will improve the morphological expression of this algorithm. In the morphological meanings, the arithmetic difference devotes to morphological gradient [15].

Storage flag. The storage operation was firstly introduced in [14] represents whether the result of this gene should be stored or not. When the finite automaton parses a chromosome, the input of current gene is the previous one's result and the output is the input of next one. Therefore, the intermediate result is abandoned. The storage operation means whether memorizes the intermediate result, which can be used in subsequent logical operations.

Logical operation. The logical operator flag represents which logical operation will be applied to current gene. Four logical operations are defined in this algorithm, *AND, OR, NOT* and *XOR*. Some logical operations need two operands: one operand is current morphological operation result, and the other one is the stored intermediate result mentioned above. When being applied on the binary images, the logical operations are bitwise. There are some changes when the logical operations are applied to gray scale images [15].

Switch operation. In this paper, we bring forward a new operation— "switch", which helps to keep two operation sequences in one chromosome. When the automaton parses a chromosome, there is a register named "backup", which is initiated with the source image and changed by the switch flag. If the switch flag is true, the input of current gene is exchanged with the backup. Therefore, the backup is brought foreground to be processed in this section, and the current input is stored as the backup. The switch operation accompanied with storage operation can express two operation sequences in one chromosome. Actually, many morphological operations are hardly realized in one process [15]. Because of two operation processes contained in one chromosome, this algorithm can express most of the morphological operations with basic erosion and dilation [15].

Pattern of structural elements. Erosion and dilation are basic morphological operations [5][15], which depend on the pattern of SE. The pattern space of SE is exponentially increased along with its size . For example, a SE whose size is $n \times n$ has $2^{n \times n}$ type of patterns. Therefore, we define the patterns manually. We use the concept of *regular and irregular* SEs same as [13], but with some differences: regular(irregular) SEs are defined which are all symmetrical(asymmetrical, vice versa). Three sizes of SEs are used in our algorithm, 3×3, 5×5 and 7×7. Each size has two styles, regular and irregular, and each style has 16 patterns. Patterns of 3×3 SEs are shown in Fig 2.

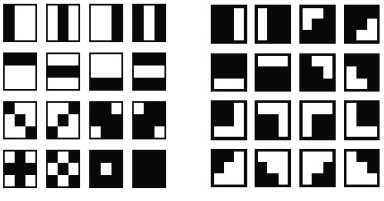


Fig. 2. Patterns of 3×3 *regular & irregular* structural elements

Fitness function. The objective fitness function $F(0 \leqslant F \leqslant 1)$ is known as similarity as the correlation coefficient between a processed image and the goal [18],

$$F = \frac{(f \cdot g)}{\sqrt{(g \cdot g)} \times \sqrt{(f \cdot f)}}$$

where f and g are two binary images of size $M \times N$ and

$$(f \cdot g) = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} f(i,j) \cdot g(i,j),$$

image f is the processed result, and image g is the goal.

Since in a binary image the white pixels represent objects, this is a reasonable choice of fitness function. The optimum is $F = 1$ when all the pixels match. The worst case is $F = 0$ when none of the pixels match [18].

2.3 Evolution Strategy

We define a structural evolution strategy in this algorithm. This algorithm initiates 1024 chromosomes at the beginning which are all single genes. The chromosome will grow in evolution and its length is limited to no more than 20 genes. The selection rate is 0.3, the mutation rate is 0.3, the crossover rate is 0.4, max generations is 300.

Structural mutation. Each gene has 14 valid bits and the random variant range is $2^{14} = 16348$. Therefore, it is hard to evolve effectively if the gene are mutated randomly. We bring forward a structural mutation strategy. All these 14 bits in one gene are divided into three parts according to their function: flow control, logical operation and morphological operation. These three parts are mutated separately, which makes the mutation position well–distributed.

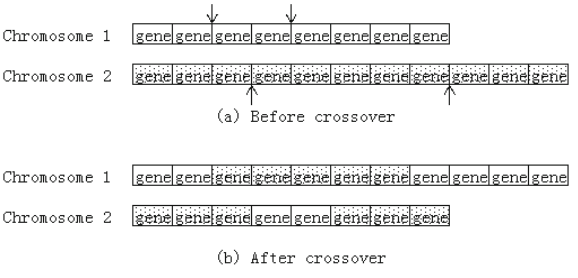


Fig. 3. Crossover

Structural crossover. The genetic operation of crossover(sexual recombination) allows new individuals to be created [9]. We use a structural crossover in this algorithm. The basic unit of crossover is gene and crossover in the middle of a gene is not allowed. The goal of the structural crossover is to exchange a bunch of genes of two participated chromosomes. The crossover generates new individuals which may have different lengths from their parents. There are two random variants in each chromosome which participate crossover, the start position and the end position (Fig. 3). And these variants result in three different meanings:

- If the amount of one of the switched parts is zero, this means that a section of the counterpart should be inserted in it.
- If both amounts of switched parts are not zero, this means to switch sections of two chromosomes, as shown in Fig 3(b).
- If the insert position is the beginning or the end of a chromosome, this means to extend this chromosome.

3 Experimental Results

We conduct three experiments to verify our algorithm. First, we apply this algorithm to two binary image analysis experiments: artificial objects extraction and OCR music sheet analysis. These tests have been used by former researchers and we compare the effectiveness of our algorithm with other other works. Next, we apply this algorithm to gray scale image enhancement to check the effects.

3.1 Experiment on Artificial Binary Images

We use this algorithm to obtain valid objects in an artificial image composed by four features: squares, disks, rings and stars, same as Quintana et al. in [13]. Each target image has one type of the four features with randomly distributed position with size of 640×480. All four target images are overlaid to obtain the source image(Fig 4). The features randomly distributed in the source image may overlap, which makes their detection more difficult.

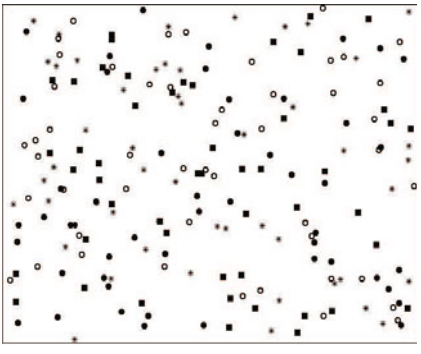


Fig. 4. Source image of artificial dataset

Training course. We select a small area from the source image which contains all four features as the source of the training set. The same areas on the target images are also selected as target images in the training set, as shown in Fig 5.

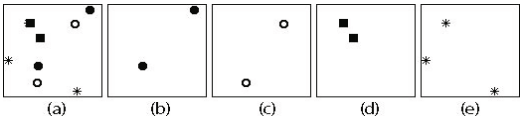


Fig. 5. Selected area of source and target images. a: source image, b c d e: target images

Testing course. The methods of four features detection are obtained in the training course by our algorithm. We apply them on the whole image, and four features are extracted separately. Executing these procedures for 10 times, we get the mean performance of our algorithm (Table. 2). The criterion is the fitness values of processed results and the target images. Except the extraction of rings, our algorithm is better than Quintana's on squares, disks and stars. Furthermore, Quintana et al. use a Linux cluster with one master node (CPU dual Intel Xeon 2 GHz, 2 GB Memory) and 22 client nodes (CPU Dual Athlon MP 1900+, 1.6 GHz, 1 GB Memory) in their experiments [13]. Our algorithm performs on a PC (CPU Intel Core Duo T9400 2.53 GHz, 4 GB Memory). Even on the most time consuming training of star, our algorithm get the result in a few minutes. This indicates our algorithm is more effective and fewer computational load.

Table 2. Performance comparison of our GP algorithm with Quintana et al.'s

Feature	Our algorithm	Quintana et al.'s
Disks	**0.9822**	0.868
Rings	0.8583	**0.906**
Squares	**0.9900**	0.870
Stars	**0.9590**	0.922

3.2 Experiment on OCR Music Sheet

We apply our GP algorithm to another type of binary image — the OCR music sheet, which was used by Yoda et al. in [18] to explore the automatic acquisition of morphological procedures by GA. We capture this music sheet from the original paper. The object of this test is to extract the four features, heads, hooks, staff lines and stems from the music sheet. Similarly, a small typical area is selected from the original image as the source image which contains all the four features. Four target images are obtained by manually calibrating, and each of which only contains one feature of four types. In the training cource of heads detection, Yoda et al. get the final fitness value of 0.963 [18], and our algorithm gets 0.9662— more accurate. The best fitness values of heads, staffs lines and stems all converge fast and stable. The best fitness value of hooks goes through a longer period, and converges to 0.9780. When we apply the methods obtained to whole music sheet, all four features are detected accurately. Experimental results also support that the efficiency and effectiveness of our algorithm are all better than Yoda et al.'s.

3.3 Experiment on Gray Scale Image

In this process, we apply this algorithm to the enhancement of gray scale image. There are more 2000 low quality finger vein images collected by an infrared CCD device, which contains 400 fingers, and each finger has 4 to 6 images. We want to use this algorithm to enhance these images and expect to extract more accurate vein features. We randomly choose four images from data set as learning samples and manually calibrate the corresponding feature images as learning goals, and all these images compose four pair of "training set". It is difficult to evaluate the effects of gray scale image enhancement. Because of the purpose of image enhancement is to get more accurate finger vein features, we use an threshold and thinning method to get the skeleton feature results of enhanced images which will be used in the evaluation. The GP algorithm runs for four times with these training set, each time with a pair of source and target, and produces four methods.

We get four operation sequences in the training course, namely morphological image analysis method(MAIM $1 \sim 4$, corresponding to four pairs of training set). Fig. 6 illustrates the four pairs of training set and the training results, images in column *a* are original images, in *b* are corresponding learning goals, in *c* are the learning results followed by threshold and thinning, and in *d* are the results by using the contrastive method which is processed by mean filtering with threshold and thinning. From the comparison, the generated methods enhance the images and more accurate features are extracted.

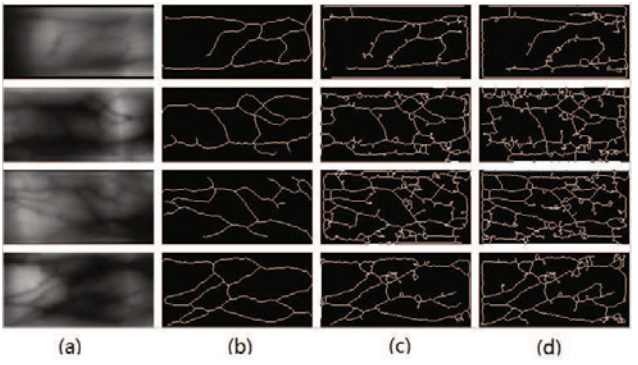


Fig. 6. Visual effect comparison for different learning pairs: (a) four original learning samples; (b) corresponding learning goals; (c) learning results of MAIM $1 \sim 4$ with threshold and thinning;(d) results obtained by mean filter with threshold and thinning

After, the obtained methods are applied to the whole data set and we examine the effects of the enhancement and verify the features obtained from enhanced images in the application of identity authentication. We test this process with combination of different number and order of MAIMes. We use the false acceptance rate (FAR) to evaluate our proposed algorithm, which is the most commonly used measure of identity authentication, the fraction of access attempts by an un–enrolled individual that are nevertheless deemed a match [17]. The classifier is Nearest Neighbor [4] and the experimental strategy is Leave–One–Out. We use the Modified Hausdorff Distance(MHD) [3][8] to measure the similarity of two images. The results are shown in Fig. 7. Bar 1 is FAR result of the contrastive method and the others are FARs of different combined MAIMes.

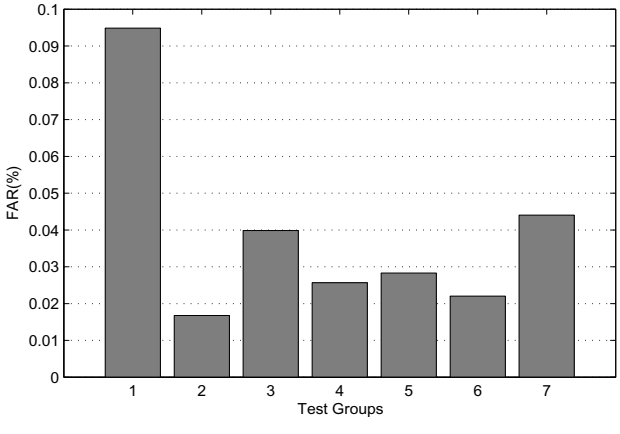


Fig. 7. Comparison of false acceptance rate for test groups and the control group. Bar 1, FAR result of control group. Bar 2, FAR result of MAIM-1,2,3,4. Bar 3, FAR result of MAIM-4,3,2,1. Bar 4, FAR result of MAIM-1,2,3. Bar 5, FAR result of MAIM-1,3,4. Bar 6, FAR result of MAIM-1,2,4. Bar 7, FAR result of MAIM-4,2,1.

Fig. 7 indicates that the enhancement methods produced by our algorithm all decrease the FARs. The most remarkable is the Bar 2 which FAR result of images are enhanced by MAIM-1,2,3,4(sequence-dependent) reduces the FAR by about 8%.

4 Conclusion

For a long time, researchers explore GA and GP approach in searching automatically produce morphological image processing methods. But the heavy computational load of evolution prevents the application of this approach. In this paper, we propose an applicable genetic programming approach to solve the problems of binary image analysis and gray scale image enhancement. It has strong ability of generalization, and shows robustness in experiments.

References

1. Ballerini, L., Franzén, L.: Genetic optimization of morphological filters with applications in breast cancer detection. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 250–259. Springer, Heidelberg (2004)
2. Daida, J.M., Hommes, J.D., Bersano-Begey, T.F., Ross, S.J., Vesecky, J.F.: Algorithm discovery using the genetic programming paradigm: extracting low-contrast curvilinear features from sar images of arctic ice, pp. 417–442 (1996)
3. Dubuisson, M.-P., Jain, A.: A modified hausdorff distance for object matching. In: Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 1, pp. 566–568 (October 1994)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, Hoboken (2000)
5. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
6. Harvey, N., Marshall, S.: The use of genetic algorithms in morphological filter design. Signal Processing: Image Communication 8(17), 55–71 (1996)
7. Hong, J.-H., Cho, S.-B., Cho, U.-K.: A novel evolutionary approach to image enhancement filter design: Method and applications. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39(6), 1446–1457 (2009)
8. Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(9), 850–863 (1993)
9. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge (1992)
10. Langdon, W.B., Poli, R., McPhee, N.F., Koza, J.R.: Genetic programming: An introduction and tutorial, with a survey of techniques and applications. In: Fulcher, J., Jain, L.C. (eds.) Computational Intelligence: A Compendium. Studies in Computational Intelligence, vol. 115, pp. 927–1028. Springer, Heidelberg (2008)
11. Munteanu, C., Rosa, A.: Gray-scale image enhancement as an automatic process driven by evolution. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 34(2), 1292–1298 (2004)
12. Poli, R.: Genetic programming for image analysis. In: GECCO 1996: Proceedings of the First Annual Conference on Genetic Programming, pp. 363–368. MIT Press, Cambridge (1996)

13. Quintana, M.I., Poli, R., Claridge, E.: On two approaches to image processing algorithm design for binary images using GP. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoWorkshops 2003. LNCS, vol. 2611, pp. 422–431. Springer, Heidelberg (2003)
14. Quintana, M.I., Poli, R., Claridge, E.: Morphological algorithm design for binary images using genetic programming. Genetic Programming and Evolvable Machines 7(1), 81–102 (2006)
15. Soille, P.: Morphological Image Analysis: Principles and Applications. Springer-Verlag New York, Inc., Secaucus (2003)
16. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 303–311. Morgan Kaufmann Publishers Inc., San Francisco (1993)
17. Weaver, A.: Biometric authentication. Computer 39(2), 96–97 (2006)
18. Yoda, I., Yamamoto, K., Yamada, H.: Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms. Image Vision Comput. 17(10), 749–760 (1999)

Optimizing Single-Source Capacitated FLP in Fuzzy Decision Systems

Liwei Zhang, Yankui Liu*, and Xiaoqing Wang

College of Mathematics & Computer Science,
Hebei University Baoding 071002, Hebei, China
zhangliwei09@139.com, yliu@hbu,edu.cn, wxqwyzy@163.com

Abstract. This work develops a new fuzzy version of single-source capacitated facility location problem (FLP), in which a set of capacitated facilities is selected to provide service to demand points with possibility distributions at the minimal total cost. Since the proposed FLP includes credibility service level constraints and 0–1 decision variables, its solution method is a challenge issue for research, and usually relies on metaheuristics and approximation approach. However, for frequently used trapezoidal, Gamma and Normal fuzzy demands, the FLPs are equivalent to deterministic 0-1 programming problems. As a consequence, the equivalent 0-1 programming problems can be solved by general purpose software or conventional optimization algorithms. At the end of this paper, we demonstrate the developed modeling idea via numerical experiments.

Keywords: Capacitated facility location problem, Fuzzy programming, Credibility service level, Integer programming.

1 Introduction

The single-source capacitated facility location problem (FLP) studies an FLP and a generalized assignment problem between a set of demand points and a set of capacitated facilities via some decision criterion. In the literature, most research focused on deterministic single-source capacitated FLPs, which are also related to uncapacitated FLPs. The interested reader may refer to the review on FLPs [1]. As for the successful solution approaches to deterministic single-source capacitated FLPs, Ahujia et al. [2] proposed a multi-exchange search with heuristic on specially designed dynamic single-source capacitated FLP networks; Holmberg [3] designed an exact branch-and-bound method with a Lagrangian heuristic to solve single-source capacitated FLP; Tragantalerngsak [4] developed a Lagrangian relaxation-based branch-and-bound algorithm for two-echelon FLP; Cortinhal and Captivo [5] applied a tabu search with classical shift and swap moves for single-source capacitated FLP, and Korupolu, Plaxton and Rajaraman [6] suggested a simple local search heuristic for the capacitated FLP in which the service costs obey triangular inequality.

* Corresponding author.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 559–567, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Recently, some researchers have paid their attention to stochastic single-source FLPs. For example, Zhou and Liu [7] presented expected value model, chance-constrained programming and dependent-chance programming for capacitated FLP with stochastic demands, and integrated network simplex algorithm, stochastic simulation and genetic algorithm for solving these stochastic models; Lin [8] considered a stochastic single-source capacitated FLP, and designed a hybrid heuristic of Lagrangean relaxation within a branch-and-bound framework to find upper and lower bounds for this problem. With the development of fuzzy set and uncertainty theories [9,10,11,12,13], some researchers realized the importance of fuzzy uncertainty in decision systems, and applied fuzzy theory to FLPs. Zhou and Liu [14] proposed fuzzy expected cost minimization model, fuzzy α-cost minimization model, and credibility maximization model according to different decision criteria, and designed some hybrid intelligent algorithms for solving these models; Wen and Iwamura [15] considered the FLP with uncertainties, presented an α-cost model under the Hurwicz criterion with fuzzy demands, and integrated the simplex algorithm, fuzzy simulation and genetic algorithm to solve this model. As for two-stage fuzzy FLPs, Liu and Zhu [16], Liu and Tian [17], Shen and Liu [18] developed three classes of two-stage models based on minimum-risk and value-at-risk decision criteria, and discussed their approximation methods as well as the convergence of the methods about optimal solutions and optimal objective values, and Liu [19] presented a new class of two-stage fuzzy random minimum risk problem based on mean chance theory [12], and applied the developed optimization method to the capacitated FLP with fuzzy random demands. In this paper, we consider a new fuzzy version of single-source capacitated FLP, in which a set of capacitated facilities is selected to provide service to demand points with possibility distributions at the minimal total cost.

The plan of this paper is as follows. In Section 2, we formulate a new fuzzy single-source capacitated FLP with service level requirements. In Section 3, we discuss the equivalent programming problems when demands are characterized by trapezoidal, Gamma and Normal fuzzy variables. In Section 4, we present one numerical example to illustrate the developed modeling idea and effectiveness of the proposed methods. Section 5 concludes the paper.

2 Problem Formulation

In this section, we will present a new fuzzy single-source capacitated FLP with service level requirements. The model attempts to minimize the total system costs, including the fixed cost of facilities and the transportation cost. To model the problem, we adopt the following indices and parameters.

Indices:

I is the index set of facilities, $i = 1, 2, \ldots, m$, and J is the index set of customers, $j = 1, 2, \ldots, n$.

Decision variables:

$$y_i = \begin{cases} 1, \text{ if the facility } i \text{ is opened} \\ 0, \text{ otherwise,} \end{cases} \tag{1}$$

and

$$X_{ij} = \begin{cases} 1, \text{ if the customer } j \text{ is served by facility } i \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

Parameters:

f_i is the fixed cost of facility i; c_{ij} is the transportation cost from facility i to customer j; ξ_j is the fuzzy demand for customer j; Q_i is the maximum capacity of facility i, and α_i is the prescribed credibility service level of facility i.

Using the notation above, we present a new fuzzy single-source capacitated FLP, which is formally built as follows:

$$\begin{cases} \min Z = \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij} \\ \text{s.t.} \ \sum_{i=1}^{m} X_{ij} = 1, j = 1, 2, \ldots, n \\ \quad X_{ij} \le y_i, \quad i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \\ \quad \text{Cr}\left\{\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\right\} \ge \alpha_i, i = 1, 2, \ldots, m \\ \quad y_i, X_{ij} = 0, 1, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n, \end{cases} \tag{3}$$

where Cr is the credibility measure defined in [20]. The objective of problem (3) includes the fixed cost $\sum_{i=1}^{m} f_i y_i$ that facilities are opened, and the transportation cost $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij}$ from opened facilities to customers. As a consequence, the objective is to minimize the following total cost:

$$\sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij}. \tag{4}$$

The constraints of problem (3) consist of the following several items:

I: The constraints

$$\sum_{i=1}^{m} X_{ij} = 1, j = 1, 2, \ldots, m \tag{5}$$

ensure that each customer is served by a single facility.

II: The constraints

$$X_{ij} \le y_i, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \tag{6}$$

indicate a facility must be set up if a demand node is assigned.

III: The quantity Q_i in (3) is the maximum service ability of facility i, and $\alpha_i (0 < \alpha_i \le 1)$ is a prescribed credibility service level. The credibility service level to the ith customer is

$$\text{Cr}\left\{\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\right\} \ge \alpha_i, i = 1, 2, \ldots, m. \tag{7}$$

The constraints impose the credibility of fuzzy event that the demands of all customers are less than the maximal service ability at each facility i is more than the predetermined service level requirement.

3 Handling Credibility Service Level Constraints

Problem (3) is a 0-1 programming subject to credibility constraints. To solve problem (3) effectively, this section will discuss how to turn the credibility service level constraints into their equivalent deterministic forms.

First, we consider the case of trapezoidal fuzzy demand. Let ξ_j be mutually independent trapezoidal fuzzy variables $(r_{j1}, r_{j2}, r_{j3}, r_{j4})$. According to the independence of the fuzzy variables [21], one has

$$\sum_{j=1}^{n} \xi_j X_{ij} = \left(\sum_{j=1}^{n} r_{j1} X_{ij}, \sum_{j=1}^{n} r_{j2} X_{ij}, \sum_{j=1}^{n} r_{j3} X_{ij}, \sum_{j=1}^{n} r_{j4} X_{ij} \right), \quad (8)$$

which is also a trapezoidal fuzzy variable. In this case, the credibility service level constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\}$ can be transformed into an equivalent deterministic constraint, which is stated in the following theorem.

Theorem 1. *For any given $\alpha_i \in (0, 1]$, we have:*

(i) When $\alpha_i \ge 0.5$, the constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\} \ge \alpha_i$ if and only if

$$(2 - 2\alpha_i) \sum_{j=1}^{n} r_{j3} X_{ij} + (2\alpha_i - 1) \sum_{j=1}^{n} r_{j4} X_{ij} - Q_i y_i \le 0. \quad (9)$$

(ii) When $\alpha_i < 0.5$, the constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\} \ge \alpha_i$ if and only if

$$(1 - 2\alpha_i) \sum_{j=1}^{n} r_{j1} X_{ij} + 2\alpha_i \sum_{j=1}^{n} r_{j2} X_{ij} - Q_i y_i \le 0. \quad (10)$$

Proof. On the basis of the possibility distribution of fuzzy variable $\sum_{j=1}^{n} \xi_j X_{ij}$, and the definition of credibility function $\mathrm{Cr}\{\xi \le x\}$, one has

$$\mathrm{Cr}\{\textstyle\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\} = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} r_{j4} X_{ij} \le Q_i y_i \\ 1 - \frac{\sum_{j=1}^{n} r_{j4} X_{ij} - Q_i y_i}{2(\sum_{j=1}^{n} r_{j4} X_{ij} - \sum_{j=1}^{n} r_{j3} X_{ij})}, & \text{if } \sum_{j=1}^{n} r_{j3} X_{ij} \le Q_i y_i \le \sum_{j=1}^{n} r_{j4} X_{ij} \\ \frac{1}{2}, & \text{if } \sum_{j=1}^{n} r_{j2} X_{ij} \le Q_i y_i \le \sum_{j=1}^{n} r_{j3} X_{ij} \\ \frac{\sum_{j=1}^{n} r_{j1} X_{ij} - Q_i y_i}{2(\sum_{j=1}^{n} r_{j1} X_{ij} - \sum_{j=1}^{n} r_{j2} X_{ij})}, & \text{if } \sum_{j=1}^{n} r_{j1} X_{ij} \le Q_i y_i \le \sum_{j=1}^{n} r_{j2} X_{ij} \\ 0, & \text{otherwise.} \end{cases}$$

Hence, if $\alpha_i \ge 0.5$ for $i = 1, 2, \ldots, m$, then $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \le Q_i y_i\} \ge \alpha_i$ is equivalent to

$$1 - \frac{\sum_{j=1}^{n} r_{j4} X_{ij} - Q_i y_i}{2(\sum_{j=1}^{n} r_{j4} X_{ij} - \sum_{j=1}^{n} r_{j3} X_{ij})} \ge \alpha_i,$$

i.e., $(2 - 2\alpha_i) \sum_{j=1}^{n} r_{j3} X_{ij} + (2\alpha_i - 1) \sum_{j=1}^{n} r_{j4} X_{ij} - Q_i y_i \le 0$.

On the other hand, when $\alpha_i < 0.5$ for $i = 1, \ldots, m$, the service level constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \leq Q_i y_i\} \geq \alpha_i$ is equivalent to

$$\frac{\sum_{j=1}^{n} r_{j1} X_{ij} - Q_i y_i}{2(\sum_{j=1}^{n} r_{j1} X_{ij} - \sum_{j=1}^{n} r_{j2} X_{ij})} \geq \alpha_i,$$

i.e., $(1 - 2\alpha_i) \sum_{j=1}^{n} r_{j1} + 2\alpha_i \sum_{j=1}^{n} r_{j2} X_{ij} - Q_i y_i \leq 0$. The proof of the theorem is complete.

In practical decision problem, the facilities with large capacities are always required to have high service levels. As a consequence, problem (3) can be converted into the following programming problem:

$$\begin{cases} \min Z = \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij} \\ \text{s.t.} \ \sum_{i=1}^{m} X_{ij} = 1, j = 1, 2, \ldots, n \\ \quad X_{ij} \leq y_i, \quad i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \\ \quad (2 - 2\alpha_i) \sum_{j=1}^{n} r_{j3} X_{ij} + (2\alpha_i - 1) \sum_{j=1}^{n} r_{j4} X_{ij} - Q_i y_i \leq 0, \\ \quad i = 1, 2, \ldots, m \\ \quad y_i, X_{ij} = 0, 1, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n. \end{cases} \tag{11}$$

We now consider the case of Gamma fuzzy demand. Let ξ_j's be Gamma fuzzy variables with parameters λ_j such that their possibility distributions are $\mu_{\xi_j}(x) = (x/\lambda_j r))^r \exp(r - x/r)$, where $x \geq 0, \lambda_j \in \Re$, and r is a fixed constant. When these Gamma fuzzy variables are mutually independent, their sums $\sum_{j=1}^{n} \xi_j X_{ij}$ are also Gamma fuzzy variables with parameters $\sum_{j=1}^{n} \lambda_j X_{ij}, i = 1, \ldots, m$. In this case, we have the following results about credibility service level constraints.

Theorem 2. *For any given $\alpha_i \in (0, 1]$, we have:*

(i) When $\alpha_i \geq 0.5$, the constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \leq Q_i y_i\} \geq \alpha_i$ if and only if

$$\ln 2(1 - \alpha_i) - r \ln Q_i y_i + r \ln \sum_{j=2}^{n} \lambda_j X_{ij} r + Q_i y_i / \sum_{j=1}^{n} \lambda_j X_{ij} - r \leq 0. \tag{12}$$

(ii) When $\alpha_i < 0.5$, the constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \leq Q_i y_i\} \geq \alpha_i$ if and only if

$$\ln 2\alpha_i - r \ln Q_i y_i + r \ln \sum_{j=1}^{n} \lambda_j X_{ij} r + Q_i y_i / \sum_{j=1}^{n} \lambda_j X_{ij} - r \leq 0. \tag{13}$$

According to Theorem 2, if service level $\alpha_i \geq 0.5$, then problem (3) can be turned into the following deterministic equivalent 0–1 programming problem:

$$\begin{cases} \min Z = \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij} \\ \text{s.t.} \ \sum_{i=1}^{m} X_{ij} = 1, j = 1, 2, \ldots, n \\ \quad X_{ij} \leq y_i, \quad i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \\ \quad \ln 2(1 - \alpha_i) - r \ln Q_i y_i + r \ln \sum_{j=2}^{n} \lambda_j X_{ij} r \\ \quad + Q_i y_i / \sum_{j=1}^{n} \lambda_j X_{ij} - r \leq 0, i = 1, 2, \ldots, m \\ \quad y_i, X_{ij} = 0, 1, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n. \end{cases} \tag{14}$$

Before ending this section, we discuss the case when Normal possibility distributions included in credibility constraints. Let ξ_j's be Normal fuzzy variables with possibility distributions $\mu_{\xi_j}(x) = \exp(-(x-a_j)^2/\sigma_j^2)$, $a_j \in \Re, \sigma_j > 0$. Note that the sum of mutually independent Normal fuzzy variables is also a Normal fuzzy variable. That is, $\sum_{j=1}^{n} \xi_j X_{ij}$ is a Normal fuzzy variable with parameter $(\sum_{j=1}^{n} a_j X_{ij}, \sum_{j=1}^{n} \sigma_j X_{ij})$, which leads to the following result:

Theorem 3. *For any given $\alpha_i \in (0, 1]$, we have:*

(i) When $\alpha_i \geq 0.5$, the constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \leq Q_i y_i\} \geq \alpha_i$ if and only if

$$Q_i y_i - \sum_{j=1}^{n} a_j X_{ij} - \sqrt{-\sum_{j=1}^{n} \sigma_j^2 \ln 2(1-\alpha_i)} \geq 0. \tag{15}$$

(ii) When $\alpha_i < 0.5$, the constraint $\mathrm{Cr}\{\sum_{j=1}^{n} \xi_j X_{ij} \leq Q_i y_i\} \geq \alpha_i$ if and only if

$$Q_i y_i - \sum_{j=1}^{n} a_j X_{ij} + \sqrt{-\sum_{j=1}^{n} \sigma_j^2 \ln 2\alpha_i} \geq 0. \tag{16}$$

Therefore, in the case when credibility service level $\alpha_i \geq 0.5$, problem (3) can be turned into the following equivalent 0–1 programming problem:

$$\begin{cases} \min Z = \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij} \\ \text{s.t. } \sum_{i=1}^{m} X_{ij} = 1, j = 1, 2, \ldots, n \\ \quad X_{ij} \leq y_i, \quad i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \\ \quad Q_i y_i - \sum_{j=1}^{n} a_j X_{ij} - \sqrt{-\sum_{j=1}^{n} \sigma_j^2 \ln 2(1-\alpha_i)} \geq 0, 1 \leq i \leq m \\ \quad y_i, X_{ij} = 0, 1, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n. \end{cases} \tag{17}$$

So far, we have discussed the issue about handling credibility service level constraints. In the cases when the demands are mutually independent trapezoidal, Gamma and Normal fuzzy variables, problem (3) can be turned into its equivalent 0–1 programming problem (11), (14), and (17), respectively. Therefore, conventional optimization method like branch-and-bound, or general purpose software can be employed to solve the equivalent 0–1 programming problems (11), (14), and (17). In the next section, we will perform numerical experiments to demonstrate the developed modeling idea.

For general problem (3), the demands included are not necessarily independent fuzzy variables, or they have general possibility distributions instead of the special cases discussed above. To handle the general cases, we may combine approximation method [22] and conventional optimization methods to solve problem (3).

4 Numerical Experiments

To illustrate the developed modeling idea and effectiveness of the proposed methods, we have performed a number of numerical experiments. For the sake of presentation, this section provides some of them via one numerical example about

fuzzy single-source capacitated FLP with 10 facilities and 20 customers. The set of data in this problem is collected in Table 1 including the generalized credibility service level α_i, the maximum capacities of facilities Q_i, the fixed opened costs f_i, and the demands of customers ξ_j; while transportation costs from facility i to customer j are provided in Table 2.

Table 1. The Set of Data in Numerical Experiments

$m \times n$	10×20				
i	1	2	3	4	5
	6	7	8	9	10
f_i	10	12	12	10	14
	14	12	16	12	15
Q_i	80	90	95	90	92
	89	94	92	91	96
α_i	0.85	0.85	0.85	0.85	0.85
	0.85	0.85	0.85	0.85	0.85
j	1	2	3	4	5
	6	7	8	9	10
	11	12	13	14	15
	16	17	18	19	20
ξ_j	(16,18,19,25)	(12,13,23,29)	(10,11,12,13)	(18,20,23,25)	(11,13,24,37)
	(19,23,26,29)	(23,24,32,41)	(17,20,24,34)	(18,21,46,52)	(12,24,39,41)
	(22,24,26,29)	(16,21,32,40)	(18,22,24,35)	(28,32,46,49)	(12,24,30,41)
	(20,22,26,29)	(27,31,32,40)	(17,22,24,35)	(28,31,46,49)	(12,24,27,47)

Table 2. Transportation Costs from Facility i to Customer j

c_{ij}	$j=1$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$i=1$	24	35	18	19	11	23	29	23	28	26	24	35	18	19	11	23	29	23	28	26
2	25	31	26	41	34	25	27	28	32	33	25	31	26	41	34	25	27	28	32	33
3	16	37	41	18	24	26	18	22	38	27	16	37	41	18	24	26	18	22	38	27
4	33	32	32	42	25	26	24	23	18	28	33	32	32	42	25	26	24	23	18	28
5	41	42	43	42	42	32	26	27	28	36	41	42	43	42	42	32	26	27	28	36
6	32	31	41	19	18	18	19	41	28	18	32	31	41	19	18	18	19	41	28	18
7	26	22	32	23	24	24	26	26	28	35	26	22	32	23	24	24	26	26	28	35
8	19	31	41	21	19	19	18	17	18	19	19	31	41	21	19	19	18	17	18	19
9	19	19	31	41	21	11	31	41	28	39	19	19	31	41	21	11	31	41	28	39
10	29	22	31	43	31	36	41	48	38	29	29	22	31	43	31	36	41	48	38	29

Using the given sets of data in Tables 1 and 2, we employ LINGO 8.0 software to solve the equivalent mathematical programming problem (11) with $m = 10, n = 20$, and credibility service level $\alpha_i = 0.80$. In this case, the minimum cost is 478.0000 with the following optimal solution

$X_{13} = X_{15} = X_{1,13} = X_{34} = X_{3,11} = X_{3,17} = X_{49} = X_{4,15} = X_{57} = X_{5,19} = X_{6,10} = X_{6,20} = X_{7,12} = X_{7,14} = X_{81} = X_{88} = X_{8,18} = X_{92} = X_{96} = X_{9,16} = 1$, and $y_1 = y_3 = y_4 = y_5 = y_6 = y_7 = y_8 = y_9 = 1$.

Furthermore, in order to demonstrate the influence of service level parameter to solution quality, we also compare solutions with different values of credibility level α_i. The computational results are reported in Table 3.

Table 3. Comparison of Solutions with Different Service Levels

$m \times n$	α_i	Max-Iter	Optimal Objective Values	Time (sec)
10×20	0.80	830	478.0000	1
10×20	0.85	830	481.0000	1
10×20	0.90	941	481.0000	2
10×20	0.95	25319	493.0000	5

5 Conclusions

As demand is usually uncertain in reality, this paper attempts to deal with single-source capacitated FLP with fuzzy demands and credibility service level requirements. Our main results are summarized as follows. First, we developed a new fuzzy version of single-source capacitated FLP, in which a set of capacitated facilities is selected to provide credibility service to demand points with possibility distributions at the minimal total cost. Second, we handled the credibility service level constraints in the cases when the demands are characterized by frequently used trapezoidal, Gamma and Normal fuzzy demands, and turned the original FLPs into their equivalent deterministic 0-1 linear or nonlinear programming problems. Third, some numerical experiments have been performed via one numerical example to demonstrate the developed modeling idea.

This paper studied a new fuzzy single-source FLP. The focus is on special cases when demands are trapezoidal, Gamma and Normal fuzzy variables. While some issues have been resolved, some new ones have been exposed. For general problem (3), the demands included are not necessarily independent fuzzy variables, or they have general possibility distributions. To handle the general cases, we may combine approximation method and conventional optimization methods to solve general FLP (3). So the fuzzy single-source FLP is a fertile field for research.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No.60974134, the Natural Science Foundation of Hebei Province (No.A2011201007), and the Scientific Research and Development Program of Baoding (No.10ZR002).

References

1. Klose, A., Drexl, A.: Facility Location Models for Distribution System Design. Eur. J. Oper. Res. 162(1), 4–29 (2005)
2. Ahujia, R.K., Orlin, J.B., Pallottinao, S., Scaparra, M.P., Scutella, M.G.: A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem. Manag. Sci. 50(6), 749–760 (2004)

3. Homberg, K., Hellstrand, J.: Solving the Uncapacitated Network Design Problem by a Lagrangean Heuristic and Branch-and-Bound. Oper. Res. 46(2), 247–259 (1998)
4. Tragantalerngsak, S., Holt, J., Ronnqvist, M.: An Exact Method for the Two-Echelon, Single-Source, Capacitated Facility Location Problem. Eur. J. Oper. Res. 123(3), 473–489 (2000)
5. Cortinhal, M.J., Captivo, M.: Upper and Lower Bounds for the Single Source Capacitated Location Problem. Eur. J. Oper. Res. 151(2), 333–351 (2003)
6. Korupolu, M., Plaxton, C., Rajaraman, R.: Analysis of a Local Search Heuristic for Facility Location Problems. J. Algorithms 37(1), 146–188 (2000)
7. Zhou, J., Liu, B.: New Stochastic Models for Capacitated Location-Allocation Problem. Comput. Ind. Eng. 45(1), 111–125 (2003)
8. Lin, C.K.: Stochastic Single-Source Capacitated Facility Location Model with Service Level Requirements. Int. J. Prod. Econ. 117(2), 439–451 (2009)
9. Zadeh, L.A.: Fuzzy Sets. Inf. Control 8(3), 338–353 (1965)
10. Zadeh, L.A.: Fuzzy Sets As a Basis for a Theory of Possibility. Fuzzy Sets Syst 1(1), 3–28 (1978)
11. Liu, B.: Uncertainty Theory. Springer, Berlin (2004)
12. Liu, Y.K., Liu, Z., Gao, J.: The Modes of Convergence in the Approximation of Fuzzy Random Optimization Problems. Soft Comput. 13(2), 117–125 (2009)
13. Liu, Z., Liu, Y.K.: Type-2 Fuzzy Variables and Their Arithmetic. Soft Comput. 14(7), 729–747 (2010)
14. Zhou, J., Liu, B.: Modeling Capacitated Location-Allocation Problem with Fuzzy Demands. Comput. Ind. Eng. 53(3), 454–468 (2007)
15. Wen, M., Iwamura, K.: Fuzzy Facility Location-Allocation Problem under the Hurwicz Criterion. Eur. J. Oper. Res. 184(2), 627–635 (2008)
16. Liu, Y.K., Zhu, X.: Capacitated Fuzzy Two-Stage Location-Allocation Problem. Int. J. Innov. Comput. Inf. Control 3(4), 987–999 (2007)
17. Liu, Y.K., Tian, M.: Convergence of Optimal Solutions About Approximation Scheme for Fuzzy Programming with Minimum-Risk Criteria. Comput. Math. Appl. 57(6), 867–884 (2009)
18. Shen, S., Liu, Y.K.: A New Class of Fuzzy Location-Allocation Problems and Its Approximation Method. Information 13(3A), 577–591 (2010)
19. Liu, Y.K.: The Convergent Results About Approximating Fuzzy Random Minimum Risk Problems. Appl. Math. Comput. 205(2), 608–621 (2008)
20. Liu, B., Liu, Y.K.: Expected Value of Fuzzy Variable and Fuzzy Expected Value Models. IEEE Trans. Fuzzy Syst. 10(4), 445–450 (2002)
21. Liu, Y.K., Gao, J.: The Independence of Fuzzy Variables with Applications to Fuzzy Random Optimization. Int. J. Uncertain. Fuzz. Knowl.-Based Syst. 15(suppl.2), 1–20 (2007)
22. Liu, Y.K.: Convergent Results About the Use of Fuzzy Simulation in Fuzzy Optimization Problems. IEEE Trans. Fuzzy Syst. 14(2), 295–304 (2006)

New Results on a Fuzzy Granular Space*

Xu-Qing Tang and Kun Zhang

School of Science, Jiangnan University Wuxi214122, Jiangsu Province, China
txq5139@yahoo.com.cn, txq5139@jiangnan.edu.cn

Abstract. Based on granular spaces, some relational problems with fuzzy equivalence relations is studied, and three results are obtained as follows. Firstly, the dynamic property of a fuzzy equivalence relation on its granular space is discussed. Secondly, the ordering relationship between fuzzy equivalence relations and their granular spaces is researched, and they are order-preserving. Furthermore, the collaborative clustering of fuzzy equivalence relations on granular spaces by their intersection operation is given, which the collaborative clustering derived from the fuzzy equivalence relations obtained by the intersection operation is a thinner or more precise consistent cluster. These conclusions will help us pursue an even deeper understanding of the essence of granular computing.

Keywords: Granular Computing, Fuzzy Equivalence Relation, Fuzzy Granular Space, Ordering Relationship, Collaborative Clustering.

1 Introduction

Clustering and in particular, fuzzy clustering occupy an important role in understanding data revealing their underlying structures and offering some useful insights into the general tendencies, associations and dependencies manifesting therein [1]. With the development of relationship between information granules regarded as fuzzy sets [2,3,4,5,6], the structural analysis of fuzzy sets and fuzzy clusterings on information granules have addressed, these researches help us pursue an even deeper understanding of the essence of fuzzy sets and clustering procedures.

During the past few years, a considerable number of studies have been conducted fuzzy collaborative clustering, and together with a rapid growth in the variety of applications [7,8,9,10,11,12,13]. How is to establish a collaborative clustering method? Recently, an important extension of clustering leads to the concepts of combination of several clustering results, which it is used on several differential clusterings for the same data set or clustering for several data sets. With a growing diversity of problems in which we traverse a path from data to knowledge, we encounter information processing tasks in which a vital interest is in a multifaceted perspective. In Refs.[1,2,8,9], Pedrycz *et al* conducted research on collaborative fuzzy clustering based on FCM, which it mainly concentrate on

* The research was supported in part by the Program for Innovative Research Team of Jiangnan University.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 568–577, 2011.
© Springer-Verlag Berlin Heidelberg 2011

some essential algorithmic facets of processing data. In Ref.[14], Chen and Yao proposed a multiview collaborative approach based on data operators. Besides, Miyamoto studied information fusion based on fuzzy multi-sets [15]. Esnaf *et al* proposed a fuzzy clustering-based hybrid method for a multi-facility location problem [16], and Shuu also presented the fusion problem based on fuzzy multi-attribute group decision-making with multi-granularity linguistic information [17]. These collaborative methods of structural clusters were developed to address a common problem, i.e., for given several clusterings on the same data set, how to obtain a more perfect (or better) cluster.

Recently, We also presented cluster analysis based on fuzzy quotient spaces [18], reported a direct method for solving clustering problems based on normalized metric. To better analyze clustering structures, we proposed the research on fuzzy granular spaces [19,20] based on the fuzzy quotient space theory [5,6]. In Ref.[21], we introduced granular spaces into fuzzy proximity relations, proposed the research on structural clusters of a fuzzy proximity relation, and researched the structural clustering characteristic based on a fuzzy granular space. In Ref.[22], we also presented the research on structural clustering and analysis of metric based on a granular space.

In this paper, we study some relational problems on a fuzzy granular space. It is organized as follows: in Section 2, some preliminaries on fuzzy granular space derived from a fuzzy equivalence relation are introduced; in Section 3, the dynamic property of a fuzzy equivalence relation on its granular space is discussed; in Section 4, the ordering relationship between fuzzy equivalence relations and their granular spaces is researched; in Section 5, the collaborative clustering of fuzzy equivalence relations on granular spaces is presented; Conclusions are included in Section 6.

2 Preliminaries on Fuzzy Granular Space

We will first introduce some basic concepts and lemmas, as follows.

If R is a fuzzy equivalence relation (abbreviated "FE relation") on X and satisfies the separability (i.e., $\forall x, y \in X, R(x,y) = 1 \leftrightarrow x = y$), then R is called a separable FE relation on X [19]. Let $SFE(X)$ and $FE(X)$ stand for the set of separable FE relations and the set of FE relations on X, respectively. Given $R \in FE(X)$ and $\forall \lambda \in [0,1]$, define a relation $R_\lambda : (x,y) \in R_\lambda \leftrightarrow R(x,y) \geq \lambda$ (note: R_λ is the cut equivalence relation of R on X. The set of all cut equivalence relations of R is denoted by $\Re_R(X)$, i.e., $\Re_R(X) = \{R_\lambda \mid \lambda \in [0,1]\}$. $\Re_R(X)$ is also called the equivalence relations set derived from R. The equivalence classes of R_λ is denoted by $[x]_\lambda = \{y \mid R(x,y) \geq \lambda, y \in X\}$, and $X(\lambda) = \{[x]_\lambda \mid x \in X\}$ is called a granulation corresponding to λ derived from R. The set $\{X(\lambda) \mid \lambda \in [0,1]\}$ is called a granular space on X derived by R and it is denoted by $\aleph_R(X)$. In fact, a granulation just stands for a partition of X here.

Definition 1. *Let (X,d) be a metric space. If d satisfies:*
(1) $\forall x, y \in X, 0 \leq d(x,y) \leq 1$;(2) $\forall x, y, z \in X$, there does not exist any number

within the array $\{d(x,y), d(y,z), d(z,x)\}$ *such that it is greater than the maximum of the other two numbers.*

Then d *is called a normalized ultrametric distance (NU metric) on* X*. The first above-mentioned condition (1) is also called the normalized condition, and the second above-mentioned condition (2) is called the ultrametric condition.*

If $d : X \times X \to R^+$, and d satisfies the symmetry, triangle inequality and $\forall x \in X, d(x,x) = 0$, then d is called a natural pseudo-distance on X [23]. If d is a natural pseudo-distance satisfying the normalized condition and the ultrametric condition, then d is also called a normalized and natural pseudo-ultrametric (abbreviated "NPU metric") on X. Let $NU(X)$ and $NPU(X)$ stand for the set of normalized and ultrametrics and the one of normalized and natural pseudo-ultrametrics on X, respectively.

Definition 2. *Let* $X(\lambda_1)$ *and* $X(\lambda_2)$ *be two granulations on* X*.*
(1) If $\forall x \in X$, $[x]_{\lambda_1} \subseteq [x]_{\lambda_2}$*, then granulation* $X(\lambda_2)$ *is deemed to be less fine than* $X(\lambda_1)$ *and it is denoted by* $X(\lambda_2) \leq X(\lambda_1)$*.*
(2) If $X(\lambda_2) \leq X(\lambda_1)$ *and there exists* $x_0 \in X$ *such that* $[x_0]_{\lambda_1} \subset [x_0]_{\lambda_2}$*, then* $X(\lambda_1)$ *is deemed to be more fine than* $X(\lambda_2)$ *and it is denoted as* $X(\lambda_2) < X(\lambda_1)$*.*

If $R \in SFE(X)$, then $\forall x \in X, [x]_1 = \{x\}$, i.e., $X(1) = \{\{x\} \mid x \in X\} = X$. $\forall \lambda \in [0,1], X(\lambda) \leq X(1)$, i.e., $X(1)$ (or X) is the finest granulation of X.

Lemma 1. *If* $R \in FE(X)$*, then the deriving fuzzy granular space* $\aleph_R(X)$ *is an ordered set, and it satisfies* $\forall \lambda_1, \lambda_2 \in [0,1], \lambda_1 \leq \lambda_2 \to X(\lambda_1) \leq X(\lambda_2)$*. Particularly,* $\forall \lambda_1, \lambda_2 \in D, \lambda_1 < \lambda_2 \to X(\lambda_1) < X(\lambda_2)$*, where* $D = \{R(x,y) \mid x, y \in X\}$*.*

Proof. The proof is similar to the one of Proposition 3.2 in Ref.[19], here we omitted it.

In Lemma 1, $\aleph_R(X)$ is also called the ordered fuzzy granular space derived from R. In Ref.[21], we discussed the theory of ordered fuzzy granular spaces derived from $FE(X)$, studied relationships between FE relations on X and NPU metrics on X, and given the following result.

Lemma 2. *Assume that* $R \in FE(X)$ *(or* $R \in SFE(X)$*), and there exists an one-to-one and strictly increasing mapping* g *satisfying* $g(0) = 0$ *on* $[0,1] \to [0,1]$*. Define* d *on* X*:* $\forall x, y \in X, d(x,y) = g(1 - R(x,y))$*. Then* $d \in NPU(X)$ *(or* $d \in NU(X)$*).*

In Lemma 2, the d is called a NPU (or NU) metric on X derived by R.

Lemma 3. *Assume that* $R \in FE(X)$*, that* $\aleph_R(X)$ *denotes the derived granular space from* R *and that* d *stands for the a NPU metric derived from* R*.* $\forall X(\lambda) \in \aleph_R(X)$*, define* $d_\lambda : \forall a, b \in X(\lambda), d_\lambda(a,b) = \inf\{d(x,y) | x \in a, y \in b\}$*. If* $X(\lambda) < X(1)$*, then,* d_λ *is a NU metric on* $X(\lambda)$*.*

In Ref.[21], we have discussed the relationship between the consistent clustering and the fuzzy granular spaces, and get the result that a consistent clustering of X is given for given a FE relation on X, and its granular space is just a consistent clustering of X.

3 The Dynamic Property of a FE Relation on Its Granular Space

Theorem 1. *Assume that $R \in FE(X)$, $\aleph_R(X)$ is the derived granular space from R on X. For any granulation $X(\lambda) \in \aleph_R(X)$, define a fuzzy relation R^*_λ on $X(\lambda)$: $\forall a, b \in X(\lambda), R^*_\lambda(a,b) = \sup\{R(x,y)|x \in a, y \in b\}$. If $X(\lambda) > X(1)$, then $R^*_\lambda \in SFE(X(\lambda))$.*

If the theorem holds, then R^*_λ is called the projecting FE relation of R to $X(\lambda)$.

Proof. By $R \in FE(X)$, taking $d(x,y) = 1 - R(x,y)$, $d \in NPU(X)$ and $d_\lambda \in NU(X(\lambda))$ from Lemma 2 and 3, where $d_\lambda(a,b) = \inf\{d(x,y)|x \in a, y \in b\} = 1 - \sup\{R(x,y)|x \in a, y \in b\} = 1 - R(a,b)$, $a, b \in X(\lambda)$. R^*_λ obviously satisfies the symmetry and $\forall a, b \in X(\lambda), 0 \le R^*_\lambda(x,y) \le 1$ from the definition of R^*, and
(1) $\forall a \in X(\lambda), R^*_\lambda(a,a) = \sup\{R(x,y)|x, y \in a\} = 1$.
(2)$\forall a, b \in X(\lambda), R^*_\lambda(a,b) = 1 \leftrightarrow d_\lambda(a,b) = 0 \leftrightarrow a = b$ from $d_\lambda \in NU(X(\lambda))$, i.e., R^*_λ satisfies the separability on $X(\lambda)$;
(3) $\forall a, b, c \in X(\lambda), R^*_\lambda(a,b) = 1 - d_\lambda(a,b) \ge \min\{1 - d_\lambda(a,c), 1 - d_\lambda(c,b)\} = \min\{R^*_\lambda(a,c), R^*_\lambda(c,b)\}$. So $R^*_\lambda(a,b) \ge \sup_{c \in X(\lambda)}\{\min\{R^*_\lambda(a,c), R^*_\lambda(c,b)\}\}$.
Therefore, $R^*_\lambda \in SFE(X(\lambda))$.

In Theorem 1, the R^*_λ is also called as the projective SFE relation on $X(\lambda)$ by R. Given a FE relation R, Theorem 1 states that we can get a FE relation R^*_λ on $X(\lambda)$ of X derived from R according to the projective FE relation.

Theorem 2. *Assume that $X(\lambda)$ is a granulation of X, and that $R_\lambda \in FE(X(\lambda))$ (or $R_\lambda \in SFE(X(\lambda))$). Define a fuzzy relation $\overline{R_\lambda}$ on X: $\forall x, y \in X, \overline{R_\lambda}(x,y) = R_\lambda(a,b)$, where $x \in a, y \in b, a, b \in X(\lambda)$. Then, $\overline{R_\lambda} \in FE(X)$.*

Proof. When $R \in FE(X(\lambda))$, $\overline{R_\lambda}$ obviously satisfies the symmetry and $\forall x, y \in X, 0 \le \overline{R_\lambda}(x,y) \le 1$ from the definition of $\overline{R_\lambda}$. In addition,
(1) $\forall x \in X, \overline{R_\lambda}(x,x) = R_\lambda(a,a) = 1$, where $x \in a \in X(\lambda)$;
(2) $\forall x, y, z \in X, \overline{R_\lambda}(x,y) = R(a,b) \ge \sup_{c \in X(\lambda)}\{\min\{R_\lambda(a,c), R_\lambda(c,b)\}\} = \sup_{z \in X}\{\min\{\overline{R_\lambda}(x,z), \overline{R_\lambda}(z,y)\}\}$, where $x \in a, y \in b, z \in c, a, b, c \in X(\lambda)$. So $\overline{R_\lambda}(x,y) = R(a,b) \ge \sup_{z \in X}\{\min\{\overline{R_\lambda}(x,y), \overline{R_\lambda}(z,y)\}\}$.
Therefore, $\overline{R_\lambda} \in FE(X)$.
As in the above proof, it also holds when $R_\lambda \in SFE(X(\lambda))$.

In Theorem 2, $\overline{R_\lambda}$ is also called the extending FE relation on X from the R on $X(\lambda)$. Theorem 2 shows that we can get a FE relation $\overline{R_\lambda}$ on X by the extending FE relation for given a FE relation $R_\lambda \in FE(X(\lambda))$.

Remark 1. From Theorem 1 and 2, the dynamic property of FE relations on an ordered granular space is given, i.e., given a (separable) FE relation on a granulation $X(\lambda)$, we can determine a separable FE relation on any granulation that is more coarse than $X(\lambda)$ by its projective FE relation; we can also define the FE relation on any granulation that is finer than $X(\lambda)$ by its extending FE relation.

For a given FE relation R on X, its granular space is denoted by $\aleph_R(X) = \{X(\lambda)|\lambda \in [0,1]\}$. The set, which is all extending FE relations of projective FE relations on all granulations $X(\lambda)$ (Note: it is called the *extending FE relations set*), is denoted by $\Re_R(\aleph)$, i.e., $\Re_R(\aleph) = \{\overline{R^*_\lambda}|0 \le \lambda \le 1\}$.

Definition 3. *Let $\aleph_1(X)$ and $\aleph_2(X)$ be two ordered granular spaces on X, and their granulations are marked as $X_1(\lambda)$ and $X_2(\lambda)$ ($\lambda \in [0,1]$), respectively.*
(1) If $\forall\lambda \in [0,1], X_1(\lambda) \le X_2(\lambda)$, then the granular space $\aleph_1(X)$ is deemed to be less fine than $\aleph_2(X)$ and it is denoted as $\aleph_1(X) \le \aleph_2(X)$;
(2) If $\aleph_1(X) \le \aleph_2(X)$ and there exists $\lambda_0 \in [0,1]$ such that $X_1(\lambda_0) < X_2(\lambda_0)$, then $\aleph_2(X)$ is deemed to be more fine than $\aleph_1(X)$ and it is marked as $\aleph_1(X) < \aleph_2(X)$.

Theorem 3. *Suppose that $R \in FE(X)$, and that its granular space is denoted by $\aleph_R(X) = \{X(\lambda)|\lambda \in [0,1]\}$. $\forall\lambda \in [0,1]$, R^*_λ stands for the projective FE relation derived from R on the granulation $X(\lambda)$, and $\overline{R^*_\lambda}$ is the extending FE relation of R^*_λ on X. $\aleph_{R^*_\lambda}(X(\lambda))$ and $\aleph_{\overline{R^*_\lambda}}(X)$ is the granular space derived by R^*_λ (on $X(\lambda)$) and $\overline{R^*_\lambda}$ (on X), respectively; their granulations are marked as $X_\lambda(\mu)$ and $X^*_\lambda(\mu)$, respectively ($\mu \in [0,1]$). Then, $\forall\mu \in [0,1]$, $X_\lambda(\mu) = X^*_\lambda(\mu)$.*

Proof. $\forall\lambda \in [0,1], \mu \in [0,1]$, $[x]_{\lambda,\mu}$ and $[x]^*_{\lambda,\mu}$ stands for the equivalence class corresponding to x of $X_\lambda(\mu)$ and one of $X^*_\lambda(\mu)$, respectively. By Theorem 2 and Definition 2, $\forall x \in a \in X_\lambda(\mu) \leftrightarrow a = [x]_{\lambda,\mu} = \{y|R^*_\lambda([x]_{\lambda,\mu}, [y]_{\lambda,\mu}) \ge \mu\} = \{y|\overline{R^*_\lambda}(x,y) \ge \mu\} = [x]^*_{\lambda,\mu} \leftrightarrow a \in X^*_\lambda(\mu)$, therefore, $X_\lambda(\mu) = X^*_\lambda(\mu)$.

Theorem 3 shown that the ordered granular space derived from R^*_λ is the same as the one derived from its extending FE relation.

4 The Ordering Relationship between FE Relations and Their Granular Spaces

Definition 4. *Let $R_1, R_2 \in FE(X)$.*
(1) If $\forall x, y \in X, R_1(x,y) \le R_2(x,y)$, then R_2 is deemed to be less fine than R_1 and is denoted as $R_2 \le R_1$;
(2) If $R_2 \le R_1$, and there exists $x_0, y_0 \in X$ such that $R_1(x_0,y_0) < R_2(x_0,y_0)$, then R_1 is deemed to be more fine than R_2 and is denoted as $R_2 < R_1$.

Like an amplifying or reducing scale, the "coarse-fine" relation among FE relations shows a basic property of fuzzy relations. For any two given FE relations on a universe, the smaller the relation value, the finer the fuzzy relation.

Theorem 4. *Assume that $R \in FE(X)$ (or $R \in SFE(X)$), then the extending FE relations set $\Re_R(\aleph) = \{\overline{R^*}_\lambda|0 \le \lambda \le 1\}$ is an ordered set, satisfying $\forall\lambda_1, \lambda_2 \in [0,1], \lambda_1 \le \lambda_2 \to \overline{R^*_{\lambda_1}} \le \overline{R^*_{\lambda_2}}$.*

Proof. Let $X(\lambda)$ be the granulation derived from R on X to λ, i.e., let $X(\lambda) = \{[x]_\lambda|x \in X\}$. For any $\lambda_1, \lambda_2 \in [0,1], \lambda_1 \le \lambda_2$. Then we get $X(\lambda_1) \le X(\lambda_2)$

from Lemma 1, so $\forall x, y \in X, [x]_{\lambda_2} \subseteq [x]_{\lambda_1}, [y]_{\lambda_2} \subseteq [y]_{\lambda_1}$. According to Theorem 1 and 2, $\overline{R^*_{\lambda_1}}(x,y) = R^*_{\lambda_1}([x]_{\lambda_1}, [y]_{\lambda_1}) = \sup\{R(x_1,y_1)|x_1 \in [x]_{\lambda_1}, y_1 \in [y]_{\lambda_1}\} \geq \sup\{R(x_1,y_1)|x_1 \in [x]_{\lambda_2}, y_1 \in [y]_{\lambda_2}\} = \overline{R^*_{\lambda_2}}(x,y)$. Therefore, $\overline{R^*_{\lambda_1}} \leq \overline{R^*_{\lambda_2}}$ by Definition 4.

Theorem 5. *Suppose that $R_1, R_2 \in FE(X)$ and that their deriving granular spaces are $\aleph_{R_1}(X)$ and $\aleph_{R_2}(X)$. Then, $R_1 \leq R_2 \Longleftrightarrow \aleph_{R_1}(X) \leq \aleph_{R_2}(X)$. Particularly, when $R_1, R_2 \in SFE(X)$, we have $R_1 < R_2 \Longleftrightarrow \aleph_{R_1}(X) < \aleph_{R_2}(X)$.*

Proof. From $R_1, R_2 \in FE(X)$, we denoted $\aleph_{R_i}(X) = \{X_i(\lambda)|\lambda \in [0,1]\}$, where $X_i(\lambda) = \{[x]_{i\lambda}|x \in X\}, i = 1, 2$.
"$\Longrightarrow$". $\forall \lambda \in [0,1], x, y \in X$, by the condition $R_1 \leq R_2 \leftrightarrow R_2(x,y) \leq R_1(x,y)$, i.e., $\forall x \in X, [x]_{2\lambda} = \{y|R_2(x,y) \geq \lambda\} \subseteq \{y|R_1(x,y) \geq \lambda\} = [x]_{1\lambda} \in X_1(\lambda)$. Therefore, $\aleph_{R_1}(X) \leq \aleph_{R_2}(X)$ by Definition 3.
"$\Longleftarrow$". $\forall \lambda \in [0,1], X_1(\lambda) \leq X_2(\lambda)$, i.e., $\forall x \in X, [x]_{2\lambda} \subseteq [x]_{1\lambda}$. By the extending principle of fuzzy sets, $\forall x, y \in X, R_1(x,y) = \sup_{\lambda \in [0,1]}\{\lambda \mid y \in [x]_{1\lambda}\} \geq \sup_{\lambda \in [0,1]}\{\lambda \mid y \in [x]_{2\lambda}\} = R_2(x,y)$, that is $R_1 \leq R_2$.
As in the above proof, it is easy to get $R_1 < R_2 \Longleftrightarrow \aleph_{R_1}(X) < \aleph_{R_2}(X)$ when $R_1, R_2 \in SFE(X)$, and the details are omitted here.

Theorem 5 shown that the order of $FE(X)$ (or $SFE(X)$) relations on universe X is the same as the one of their deriving granular spaces, i.e., they are order-preserving.

5 The Collaborative Clustering of FE Relations on Granular Space

Lemma 4. *$FE(X)$ (or $SFE(X)$) composes a perfect semi-order lattice under the relation "$\leq$" as defined by Definition 4. And for a given $\{R_\alpha, \alpha \in I\} \subseteq FE(X)$ (or $\subseteq SFE(X)$), define $\overline{R}$ and $\underline{R}$: $\overline{R}(x,y) = \inf_{\alpha \in I}\{R_\alpha(x,y)\}, \underline{R}(x,y) = \sup_{\alpha \in I}\{R_\alpha(x,y)\}$, then $\overline{R}$ and $\underline{R}$ is the superior and inferior of $\{R_\alpha, \alpha \in I\}$, respectively.*

Proof. The proof is similar to the one of Theorem 3.1 in [6], so we have omitted it here.

In Lemma 4, the $\overline{R}$ is just a FE relation on X by a crisp fuzzy intersection operation with $\{R_\alpha, \alpha \in I\}$, and it is denoted as $\overline{R} = \cap_{\alpha \in I} R_\alpha$.

Theorem 6. *Suppose that X_1 and X_2 are granulations on X and that $R_i \in EF(X_i)$ (or $\in SFE(X_i)$), $i = 1, 2$. $X_1 \cap X_2 = \{a \cap b \mid a \in X_1, b \in X_2\}$, define*

$$R : \forall a, b \in X_1 \cap X_2, R(a,b) = \min\{R_1(a_1,b_1), R_2(a_2,b_2)\} \quad (1)$$

where $a \subseteq a_i \in X_i, b \subseteq b_i \in X_i, i = 1, 2$. Then, $R \in FE(X_1 \cap X_2)$ (or $R \in SFE(X_1 \cap X_2)$).
If the theorem holds, R is also called a FE relation on $X_1 \cap X_2$ by the intersection operation with R_1 (on X_1) and d_2 (on X_2), and it is also denoted as $R = R_1 \cap R_2$.

Proof. When $R_i \in SFE(X_i)$ $(i = 1, 2)$, it is obvious that R satisfies the symmetry and $\forall a, b \in X_1 \cap X_2, 0 \leq R(a, b) \leq 1$ by the definition of R.
(1) $\forall a \in X_1 \cap X_2, a \subseteq a_i \in X_i, i = 1, 2, R(a, a) = 1$;
(2) $\forall a, b \in X_1 \cap X_2, R(a, b) = \min\{R_1(a_1, b_1), R_2(a_2, b_2)\} = 1 \leftrightarrow a_1 = b_1, a_2 = b_2 \leftrightarrow a = b$, i.e., R satisfies the separability on $X_1 \cap X_2$;
(3) $\forall a, b, c \in X_1 \cap X_2, a \subseteq a_i \in X_i, b \subseteq b_i \in X_i, c \subseteq c_i \in X_i$ $(i = 1, 2)$. We have $1 - R_i \in D(X_i)$ by Lemma 3 $(i = 1, 2)$. So, $1 - R_1(a_1, b_1) \leq \max\{1 - R_1(a_1, c_1), 1 - R_1(c_1, b_1)\} = 1 - \min\{R_1(a_1, c_1), R_1(c_1, b_1)\}$, i.e., $R_1(a_1, b_1) \geq \min\{R_1(a_1, c_1), R_1(c_1, b_1)\}$. Similarly, $R_2(a_2, b_2) \geq \min\{R_2(a_2, c_2), R_2(c_2, b_2)\}$. Therefore, $R(a, b) = \min\{R_1(a_1, b_1), R_2(a_2, b_2)\} \geq \min\{\min\{R_1(a_1, c_1), R_2(a_2, c_2)\}, \min\{R_1(c_1, b_1), R_2(c_2, b_2)\}\} \geq \min\{R(a, c), R(c, b)\}$. Furthermore, we have $R(a, b) \geq \sup_{c \in X_1 \cap X_2}\{\min\{R(a, c), R(c, b)\}\}$. Form (1)-(3), we have $R \in SFE(X_1 \cap X_2)$. As in the above proof, this proof also holds when $R_i \in FE(X_i)$ $(i = 1, 2)$.

Remark 2. The intersection operation in Theorem 6 is different from the crisp fuzzy intersection operation. The former is defined on different granulations of a universe, the latter is defined on the same granulation.

Theorem 7. *Suppose that X_1 and X_2 are two granulations on X and that $R_i \in FE(X_i)$ $(i = 1, 2)$. R_1^*, R_2^* stands for the extending FE relation derived from R_1 and R_2 on $X_1 \cap X_2$, respectively. Then $R_1^* \cap R_2^* = R_1 \cap R_2$, where the intersection operation of $R_1^* \cap R_2^*$ is defined by the crisp fuzzy intersection operation and the intersection operation of $R_1 \cap R_2$ is seen in Theorem 6.*

Proof. $\forall c_1, c_2 \in X_1 \cap X_2$, $c_1 \subseteq a_i \in X_i, c_2 \subseteq b_i \in X_i, i = 1, 2$, we have $R_1^* \cap R_2^*(c_1, c_2) = \inf\{R_1^*(c_1, c_2), R_2^*(c_1, c_2)\} = \inf\{R_1(a_1, b_1), R_2(a_2, b_2)\} = R_1 \cap R_2(c_1, c_2)$. Therefore, $R_1^* \cap R_2^* = R_1 \cap R_2$.

Although the intersection operation in Theorem 6 is different from the crisp fuzzy intersection operation, Theorem 7 shown that their deriving FE relations on $X_1 \cap X_2$ are the same. Furthermore, their deriving granular spaces are also the same, so they have the same consistent clusterings. From Theorem 6, 7 and Definition 4, we directly obtain the following corollary.

Corollary 1. *In Theorem 7, $R_i^* \leq R_1 \cap R_2$ $(i = 1, 2)$.*

Theorem 8. *Suppose that X_1 and X_2 are two granulations on X, and that $R_i \in FE(X_i)$, $i = 1, 2$, $R = R_1 \cap R_2$. Then, $\aleph_{R_i}(X_i) \leq \aleph_R(X_1 \cap X_2)$, where $\aleph_{R_i}(X_i)$ and $\aleph_R(X_1 \cap X_2)$ stand for the granular space derived from R_i on X_i $(i = 1, 2)$ and R (on $X_1 \cap X_2$), respectively.*

Proof. Based on Corollary 1, Theorem 4 and 5, we may easily give the proof, and we have omitted it here.

Theorem 9. *Suppose that X_1 and X_2 are two granulations on X, and that $R_i \in FE(X_i)$, $i = 1, 2$. $R \in FE(X_1 \cap X_2)$, $\aleph_{R_1}(X_1), \aleph_{R_2}(X_2)$ and $\aleph_R(X_1 \cap X_2)$ stands for the deriving granular space by R_1 (on X_1), R_2 (on X_2) and R (on $X_1 \cap X_2$), respectively. If $\aleph_{R_i}(X_i) \leq \aleph_R(X_1 \cap X_2)$ $(i = 1, 2)$, then $R_1 \cap R_2 \leq R$.*

Proof. The denoted R_1^* and R_2^* stand for the extending FE relation derived from R_1 and R_2 on $X_1 \cap X_2$, respectively. $\aleph_{R_1^*}(X_1 \cap X_2)$ and $\aleph_{R_2^*}(X_1 \cap X_2)$ is the granular space derived from R_1^* and R_2^* on $X_1 \cap X_2$, respectively. By Theorem 3 and 8, $\aleph_{R_i^*}(X_1 \cap X_2) = \aleph_{R_i}(X_i) \leq \aleph_R(X_1 \cap X_2), i = 1, 2$. Furthermore, we can derive $R_i^* \leq R, i = 1, 2$ from Theorem 5. Therefore, $R_1 \cap R_2 = R_1^* \cap R_2^* \leq R$ according to Theorem 7.

We get the following corollary directly from Theorem 7 and 9.

Corollary 2. *In Theorem 6, $R = R_1 \cap R_2$ is the superior of R_1 and R_2 on $X_1 \cap X_2$.*

Theorem 9 shown that a new structural cluster (i.e.,$\aleph_R(X_1 \cap X_2)$) may be obtained by Theorem 6 for the given two structural clusters (i.e.,$\aleph_{R_1}(X_1)$ and $\aleph_{R_2}(X_2)$). This new structural cluster is expected to satisfy $\aleph_{R_i}(X_i) \leq \aleph_R(X_1 \cap X_2)$ $(i = 1, 2)$, where it shows that the new structural cluster is finer than $\aleph_{R_1}(X_1)$ or $\aleph_{R_2}(X_2)$. By Corollary 2, the granular space $\aleph_R(X_1 \cap X_2)$ derived from R is the finest obtained from R_1 (on X_1) and R_2 (on X_2) on $X_1 \cap X_2$, and it satisfies $R_1 \cap R_2 = R_1^* \cap R_2^*$ by Theorem 7. We also gave the construction method of R in Theorem 6 and 7, where it was derived through the intersection operation of the extending FE relation R_1^* and R_1^* on $X_1 \cap X_2$, which was in turn derived from R_1 and R_1, respectively.

In fact, the results obtained above can be generalized to the collaborative clustering of an arbitrary number of clusters on universe X by Lemma 4 and Theorem 7, i.e., the following result also holds.

Theorem 10. *Assume that $\{X_\alpha \mid \alpha \in I\}$ is a granulations set on X, that R_α is a FE relation on X_α and that R_α^* is the extending FE relation of R_α on $\cap_{\alpha \in I} X_\alpha$. Then*
(1) $R = \cap_{\alpha \in I} R_\alpha \in FE(\cap_{\alpha \in I} X_\alpha)$, and $\cap_{\alpha \in I} R_\alpha = \cap_{\alpha \in I} R_\alpha^*$;
(2) $R = \cap_{\alpha \in I} R_\alpha$ is the superior of $\{X_\alpha \mid \alpha \in I\}$ on $\cap_{\alpha \in I} X_\alpha$;
(3) $\aleph_{R_\alpha}(X_\alpha) \leq \aleph_R(\cap_{\alpha \in I} X_\alpha)$ $(\alpha \in I)$, where $\aleph_{R_\alpha}(X_\alpha)$ and $\aleph_R(\cap_{\alpha \in I} X_\alpha)$ stand for the deriving granular space by R_α (on X_α) and R (on $\cap_{\alpha \in I} X_\alpha$), respectively.

Below is an example to illustrate the above theoretical results.

Example 1. Let $X_1 = \{a_1 = \{1, 2\}, a_2 = \{3, 4\}, a_3 = \{5, 6\}, a_4 = \{7, 8\}\}$ and $X_2 = \{b_1 = \{1, 2\}, b_2 = \{3, 4, 5\}, b_3 = \{6, 7, 8\}\}$ are two granulations on $X = \{1, 2, \cdots, 8\}$. $R_i \in SFE(X_i), i = 1, 2$, where $R_1 : R_1(a_i, a_i) = 1, i = 1, 2, 3, 4, R_1(a_1, a_2) = R_1(a_1, a_3) = R_1(a_1, a_4) = 0.3, R_1(a_2, a_3) = R_1(a_2, a_4) = 0.5, R_1(a_3, a_4) = 0.7$; $R_2 : R_2(b_i, b_i) = 1, i = 1, 2, 3, R_2(b_1, b_2) = 0.8, R_2(b_1, b_3) = R_2(b_2, b_3) = 0.2$. The respective granular spaces of R_1 and R_2 are as follows: $\aleph_{R_1}(X_1) = \{X_1(1) = X_1, X_1(0.7) = \{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}, X_1(0.5) = \{\{a_1\}, \{a_2, a_3, a_4\}\}, X_1(0.3) = \{X_1\}\}$; $\aleph_{R_2}(X_2) = \{X_2(1) = X_2, X_2(0.8) = \{\{b_1, b_2\}, \{b_3\}\}, X_2(0.2) = \{X_2\}\}$.

By Theorem 6, we get $X_1 \cap X_2 = \{c_1 = \{1, 2\}, c_2 = \{3, 4\}, c_3 = \{5\}, c_4 = \{6\}, c_5 = \{7, 8\}\}$. Furthermore, we get a FE relation $R = R_1 \cap R_2$ on $X_1 \cap X_2$ by formula (1), i.e., $R : R(c_i, c_i) = 1, i = 1, 2, 3, 4, 5, R(c_1, c_4) = R(c_1, c_5) =$

$R(c_2, c_4) = R(c_2, c_5) = R(c_3, c_4) = R(c_3, c_5) = 0.2, R(c_1, c_2) = d(c_1, c_3) = 0.3, d(c_2, c_3) = 0.5, R(c_4, c_5) = 0.7$. Its deriving granular space is $\aleph_R(X_1 \cap X_2) = \{X(1) = X_1 \cap X_2, X(0.7) = \{\{c_1\}, \{c_2\}, \{c_3\}, \{c_4, c_5\}\}, X(0.5) = \{\{c_1\}, \{c_2, c_3\}, \{c_4, c_5\}\}, X(0.3) = \{\{c_1, c_2, c_3\}, \{c_4, c_5\}\} X(0.2) = \{X_1 \cap X_2\}\}$, and the comparison graph of corresponding dendrogram of R_1, R_2 and R in Example 1 is shown in Fig.1, where (a), (b) and (c) stands for their dendrogram of R_1, R_2 and R, respectively. From Fig.1, we obviously have $\aleph_{R_i}(X_i) \leq \aleph_R(X_1 \cap X_2)$ $(i = 1, 2)$.

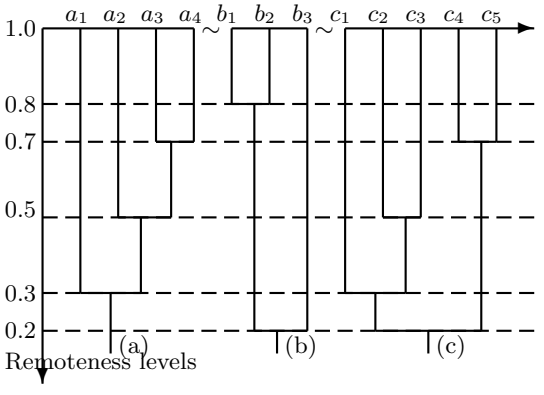


Fig. 1. The comparison graph of corresponding dendrograms in Example 1

6 Conclusions

Based on granular spaces, we study some relational problems with FE relation, and obtained results as follows: (1)The dynamic property of a FE relation on its granular space is discussed, i.e., i.e., given a (separable) FE relation on a granulation $X(\lambda)$ of X, we can determine a separable FE relation on any granulation that is more coarse than $X(\lambda)$ by its projective FE relation; we can also define the FE relation on any granulation which is finer than $X(\lambda)$ by its extending FE relation. (2)The ordering relationship between FE relations and their granular spaces is researched, and they are order-preserving. (3)The fusion on structural clusters of FE relations on granular spaces by their intersection operation is given, which the consistent cluster derived from the FE relation obtained by the intersection operation is a thinner or more precise consistent cluster.

These conclusions will help us pursue an even deeper understanding of the essence of granular computing.

References

1. Pedrycz, W., Rai, P.: Collaborative clustering with the use of Fuzzy C-Means and its quantification. Fuzzy Sets and Systems 159, 2399–2427 (2008)
2. Pedrycz, W., Rai, P.: A multifaceted perspective at data analysis: a study in collaborative intelligent agents. IEEE Trans. Systems. Man and Cybernetics-Part B: Cybernetics 38, 1062–1072 (2008)

3. Kwak, K.C., Pedrycz, W.: A design of genetically oriented linguistic model with the aid of fuzzy granulation. In: 2010 IEEE International Conference on Fuzzy Systems, July 18-23, pp. 1–6 (2010)
4. Zadeh, L.A.: Fuzzy logic = computing with words. IEEE Transactions on Fuzzy Systems 2, 103–111 (1996)
5. Zhang, B., Zhang, L.: Theory of Problem Solving and Its Applications–The theory and methods of quotient granular computing, 2nd edn. Tsinghua University Press, Beijing (2007) (in Chinese)
6. Zhang, L., Zhang, B.: The structure analysis of fuzzy sets. International Journal of Approximate Reasoning 40, 92–108 (2005)
7. Frey, B.J., Dueck, D.: Clustering by Passing messages between data points. Science 315, 972–976 (2007)
8. Devillez, A., Billaudel, P., Lecolier, G.V.: A fuzzy hybrid hierarchical clustering method with a new criterion able to find the optimal partition. Fuzzy Sets and Systems 128, 323–338 (2002)
9. Pedrycz, W.: Collaborative fuzzy clustering. Pattern Recognition Letters 23, 1675–1686 (2002)
10. Piangkh, O.S.: Analytically traceable case of fuzzy C-means clustering. Pattern Recognition 39, 35–46 (2006)
11. Tsekouras, G., Sarimreis, H., et al.: A hierarchical fuzzy clustering approach to fuzzy modeling. Fuzzy Sets and Systems 150, 246–266 (2005)
12. Sledge, I.J., Bezdek, J.C.: Relational generalizations of cluster validity indexes. IEEE Trans. Fuzzy Systems 18, 771–786 (2010)
13. Pedrycz, W., Loia, V., Senatore, S.: Fuzzy clustering with viewpoints. IEEE Trans. Fuzzy Systems 18, 274–284 (2010)
14. Chen, Y., Yao, Y.Y.: A multiview approach for intelligent data analysis based on data operators. Inf. Sci. 178, 1–20 (2008)
15. Miyamoto, S.: Information clustering based on fuzzy multi-sets. Information and Management 39, 195–213 (2003)
16. Esnaf, S., Kücükdeniz, T.: A fuzzy Clustering based hybrid method for a multi-facility location problem. J. Intell. Manuf. 20, 259–265 (2009)
17. Shuu, S.J.: Evaluating the flexibility in a manufacturing system using fuzzy multi-attribute group decision-making with multi-granularity linguistic information. Int. J. Adv. Manuf. Technol. 32, 409–421 (2007)
18. Tang, X.-Q., Zhu, P., Cheng, J.-X.: Cluster analysis based on fuzzy quotient space. Chinese Journal of Software 19, 861–868 (2008)
19. Tang, X.-Q., Zhu, P., Cheng, J.-X.: Study on fuzzy granular space based on normalized equicrural metric. Computer Science 35(4), 142–145 (2008) (in Chinese)
20. Tang, X.-Q., Zhao, J.-J., Fang, X.-S.: Research on properties of fuzzy granular space. Fuzzy Sestems and Mathematics 23(5), 70–78 (2009) (in Chinese)
21. Tang, X.-Q., Fang, X.-S., Zhu, P.: Structural clusters based on fuzzy proximity relations. Systems Engineering-Theory & Practice 30(11), 1987–1997 (2010) (in Chinese)
22. Tang, X.-Q., Zhu, P., Cheng, J.-X.: The structural clustering and analysis of metric based on granular space. Pattern Recognition 43, 3768–3786 (2010)
23. d'Amico, M., Frosini, P., Landi, C.: Natural pseudo-distance and optimal matching between reduced size function. Acta Appl. Math. 109, 527–554 (2010)

Fuzzy Integral Based Data Fusion for Protein Function Prediction

Yinan Lu, Yan Zhao, Xiaoni Liu, and Yong Quan

College of Computer Science and Technology, Jilin University
130012 Changchun, China
{luyn,quanyong}@jlu.edu.cn

Abstract. Data fusion using diverse biological data has been applied to predict the protein function in recent years. In this paper, fuzzy integral fusion based on fuzzy measure is used to integrate the probabilistic outputs of different classifiers. Support vector machines as base learners are applied to predict the functions of examples from each data source. Fuzzy density values are determined by Particle Swarm Algorithm and an improved λ-measure is used. We compare our improved fuzzy measure to typical one. The experimental results show that our method has the better results.

Keywords: Fuzzy measure; Particle Swarm Algorithm; Data fusion; Protein function prediction.

1 Introduction

Protein function prediction has become one of the main challenges in post-genome era. With the development of technology, there are large amounts of functional genomic data available, including protein-protein interaction, expression, phylogenetic profiles data, etc. How to use them becomes a major problem. Because of noise in a data set, the protein function prediction using the single source is often ineffective. Moreover, a data set can usually provide useful information only for a subset of function classes, while for others may be substantially uninformative [1]. This leads to the excellent performance in prediction of some function classes, but the poor one in others. So in recent years there have been many protein function prediction methods based on data fusion, such as Classifiers Ensemble [2][3], Artificial Neural Networks [4], Markov Random Field [5], Fusion Kernels [6], Integrated Weighted Averaging [7], Hopfield Network [8] and Simulated Annealing [9].

The weighted average is a common method in the field of information fusion. The weights in this method reflect the importance of different objects, but they can't reflect the interactions among objects. When there are interactions, the weighted average method will inevitably lead to loss of interactions information. Because of the non-additive property, fuzzy measure can reflect both of the importance of different objects and the interactions among objects. So we use Choquet fuzzy integral based on our improved λ-measure to perform the data fusion and use probabilistic support vector machines (SVM) as base learners to predict the functions of examples from each data source. The Particle Swarm Algorithm is adopted to search the optimized values of fuzzy density.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 578–586, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Currently, there are two main protein function taxonomies. One is the Functional Catalogue (FunCat) based on the Munich Information Center for Protein Sequences (MIPS) [10], the other is Gene Ontology (GO) [11]. The structure of FunCat is a tree forest and GO is a directed acyclic graph. Both of them are hierarchical. In this paper, we use FunCat to define the functions of protein.

The paper is organized as follows. In section 2, we present some definitions, the detailed description of our improved Sugeno λ-measure and how to use the Particle Swarm Algorithm to determine the fuzzy density. In section 3, we use five data sets to predict protein functions and analysis the results. The conclusions end the paper.

2 Methods

2.1 Fuzzy Measures and Choquet Integral

Definition 1. [12] Let X is a non-empty finite set $X=\{x_1, x_2,\ldots, x_n\}$, and $P(X)$ indicates the power set of X. A set function g: $P(X) \rightarrow [0, 1]$ is called a fuzzy measure g on X, if it satisfying the following axioms:

(i) $g(\phi) = 0, g(X) = 1$.

(ii) $\forall A, B \in P(X), A \subseteq B \Rightarrow g(A) \leq g(B)$.

From the above we can see that the fuzzy measures are normal and monotone. From this definition, Sugeno introduced the λ-fuzzy measure satisfying the following property:

$$g(A \cup B) = g(A) + g(A) + \lambda g(A)g(B) \ . \tag{1}$$

$$\lambda + 1 = \prod_{i=1}^{n} [1 + \lambda g(\{x_i\})], \text{ for } \lambda > -1, x_i \in X \ . \tag{2}$$

For all A, $B \in P(X)$ and $A \cap B = \phi$. In equation (2), $g(\{x_i\})$ is called fuzzy density. It can be simply denoted by g^i.

Definition 2. [13] Let g be a fuzzy measure on X and $X=\{ x_1, x_2,\ldots, x_n \}$. Without loss of generality we assume that $0 = f(x_0) \leq f(x_1) \leq \cdots \leq f(x_n) \leq 1$ and $A_i = \{x_i, x_{i+1}\ldots x_n\}$, $A_{n+1} = \phi$. The Choquet integral of f with respect to g is defined by

$$C_g(f) = \int f dg = \sum_{i=1}^{n} (f(x_i) - f(x_{i-1}))g(A_i) \ . \tag{3}$$

Note that when g is a Sugeno λ-measure, the values of $g(A_i)$ can be determined recursively by the following equation [14].

$$g(A_n) = g(\{x_n\}) = g^n \ . \tag{4}$$

$$g(A_i) = g^i + g(A_{i+1}) + \lambda g^i g(A_{i+1}) \ , \text{ for } 1 \leq i < n \ . \tag{5}$$

2.2 Our Improved Sugeno λ-Measure

We can calculate the value of λ by the equation (2), but the problem is that the value of λ is just only determined by the fuzzy density. So it cannot reflect the real relationship among the objects. In order to resolve this problem, we use Pearson correlation coefficient r instead of λ. The improved Sugeno λ-measure can be expressed as follow. For simplicity we let $s = g(A) + g(B) - r_{A,B} g(A) g(B)$.

$$g(A \cup B) = \begin{cases} s & if \quad s < 1 \\ \max(g(A), g(B)) & if \quad s \geq 1 \end{cases} \tag{6}$$

For all A, $B \in P(X)$ and $A \cap B = \phi$. If $|A|=|B|=1$, we assume that $A=\{x_a\}$ and $B=\{x_b\}$, then the value of $r_{A,B}$ is the Pearson correlation coefficient of x_a and x_b. If $|A|=1$ and $|B|>1$, we assume that $A=\{x_a\}$ and $B=\{x_{b1}, x_{b2}, \cdots, x_{bn}\}$, then $r_{A,B}$ can be estimated using the following equation (7).

$$r_{A,B} = 2 \times \prod_{x_i \in B} \frac{(1 + r_{x_a, x_i})}{2} - 1 \ . \tag{7}$$

Where r_{x_a, x_i} is the Pearson correlation coefficient of x_a and x_i. Then we can recursively calculate our improved Sugeno λ-measure using the equations (4) and (6). Because the range of $r_{A,B}$ is -1 to 1, it is easy to know that our improved Sugeno λ-measure is normal and monotone.

2.3 Determine Fuzzy Measure Based on Particle Swarm Algorithm

If the fuzzy measures are obtained, the Choquet integral is used to combine the probabilistic outputs of different support vector machines to give the final prediction. Because we use Sugeno λ-measure, it is just need to determine the fuzzy density. In this paper, the Particle Swarm Algorithm is adopted to search the optimized values of fuzzy density.

Particle Swarm Algorithm is originally attributed to Kennedy and Eberhart [15] in 1995. It has fewer parameters and it is easy to implement this algorithm. Considering a swarm of N particles, each particle's position $X_i = (x_{i1}, x_{i2}, ..., x_{id})$ represents a possible solution point in d-dimensional problem space. In this paper, the particle can be treated as fuzzy density g^i corresponding to the ith data set. In every iteration, the fitness value will be calculated when the particle flies to a new position. The iterative process will continue until the program finds the optimal solution or meets the end condition. We use the method in [16] as the update mode of the particle's flying velocity and position. The details about Particle Swarm Algorithm can be seen in [15] and [16].

Most fitness functions used to determine the fuzzy density are based on the difference of ideal integral value and the actual of integral value. But this kind of fitness function is not suitable for the problem containing the extremely unbalance between positive and negative samples (In our experiment, all of the classes have more negative samples than positive samples). When the sum of absolute differences is used as the fitness function, the particles will be extremely close to zero. Even in some classes the fuzzy density are all of zero, such as classes "30", "32" and "42". So in this paper, we use the Area Under ROC Curve (AUC) as the fitness function. Receiver Operating Characteristic (ROC) Curve is a graphical plot of the true positive rate vs. false positive rate [17]. The AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [18]. The range of AUC is 0.5 to 1. The more AUC gets close to 1, the better classification is. The fitness function AUC can be calculated as follow.

$$\text{fitness=AUC} = \frac{\sum_{i=1}^{N} \text{RANK}(R_i) - \frac{N \times (N+1)}{2}}{N \times M} . \tag{8}$$

Where N is the number of positive samples, M is the number of negative samples. $\text{RANK}(R_i)$ is the order number of Sample i after all the samples are ranked in the increasing order of probability.

Moreover, because the outputs of SVM and the results of data fusion are probabilistic, the threshold t is needed to determine the samples' class. We calculate F-measure by varying the value of t between 0 and 1 and the value of t which makes the biggest F-measure will be the value of threshold. The F-measure can be obtained by the follow equation.

$$\text{F-measure} = \frac{2 \times TP}{2 \times TP + FP + FN} . \tag{9}$$

Where TP, FP and FN are the number of true positive, false positive and false negative, respectively.

3 Experimental Results and Analysis

3.1 Data Sets and Functional Classes of Protein

The above approach is applied to predict the protein function of yeast and we chose five different types of data. Their main characteristics used in the experiments are summarized in Table 1.

For the data of Domain, we use the Pfam (protein families version 24.0) [19] official software toolkit (pfam_scan.pl) to search against the HMM (Hidden Markov Models) library of Pfam-A.hmm and Pfam-B.hmm. For each sample the presence or absence of the protein domains is stored as a binary vector.

Table 1. Data sets used in the experiments

Data set	Number of samples	Number of features
Domain	5030	4046
HMM-logE	5885	3260
Blast	5885	6101
STRING	4683	6100
Exp	6108	244

HMM-logE data are obtained by using the software toolkit HMMER [20] to calculate the log E-values of each sample against Pfam-A.hmm.

Blast data are obtained through GABLAM (Global Alignment from BLAST Local AlignMent) [21], which returns a set of pair wise sequence similarity statistics using the results from a basic BLAST search [22]. Each sample sequence is aligned with all other sequences using GABLAM. Then the negative log E-value is calculated as the score.

STRING (Search Tool for the Retrieval of Interacting Genes/Proteins version 8.3) [23] data are downloaded from STRING database. STRING is a database of known and predicted protein interactions of very high quality. For each protein pair, it has an interactive combined score S that ranges from 0 to 1. We extract the yeast data from the database and let one of the interactive protein pair as sample, the other one as the feature. Then we can get the STRING data used in our experiments.

We merge the gene expression microarray data from [24] and [25] to obtain the gene expression (Exp) data set, and select the root classes of FunCat with at least 200 positive samples (The "99 UNCLASSIFIED PROTEINS" class does not include.). We thus have 15 functional classes in our analysis. The details about the classes and the positive samples are seen in [26].

3.2 Experimental Setup and Results

The base learner of our approach is the SVM classifier. We use LIBSVM [27] software toolkit in our paper and modify the source code of LIBSVM, so that it can meet the requirements of our experiments.

First of all, we scale all the data to [0, 1] by svm-scale program in LIBSVM and find out that the performance of the classification can be improved and the computing time can be also reduced. The parameters of SVM are optimized by the modificatory grid.py in LIBSVM. Then we applied 10-fold cross-validation method to each data set. Except for the Gaussian kernel on Exp data, all others used a linear kernel.

We use particle swarm optimization toolbox [28] in MATLAB developed by Prof Brian Birge to optimize the fuzzy density. The goal of PSO is to maximize the value of the fitness function until the value is convergent. We set the number of particle swarm is 24, the number of the particle dimension is 5, the flying range of particle is [0, 1] and the maximum speed of the particle is 0.2. The other parameters are the default values of Brian Birge's program. The 10-fold cross-validation method is also used in the two data fusion methods.

Fig. 1 shows that both of the two Choquet integral fusion methods based on the typical Sugeno λ-measure and our improved Sugeno λ-measure can significantly improve the performance of the prediction. We can also find out that the performance of

the prediction based on STRING data is very good (Thirteen of the fifteen classes' AUC values ranked first in the five independent predictions.), but the performance of the prediction based on the Exp data is very poor (All of the fifteen classes' AUC values ranked last). On one hand this can prove the high quality of the STING database; on the other hand this can be a proof that the high-throughput data Exp has very high degree of noise.

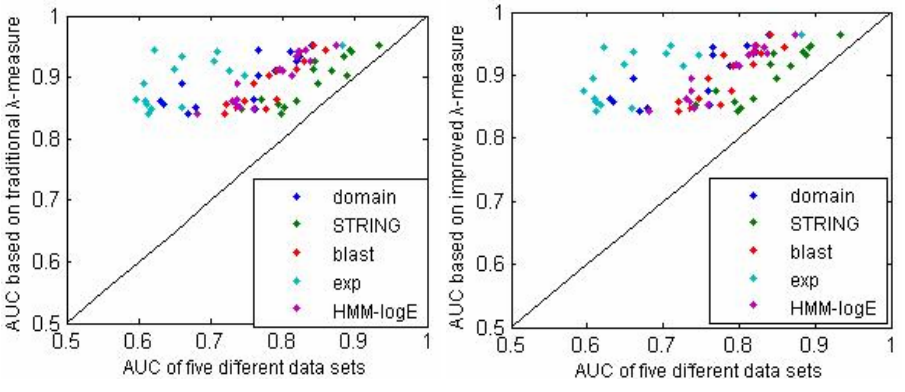


Fig. 1. Scatter-plot of AUCs of the fusion methods versus SVM predictions of five different data sets. The figure on the left is the comparison of the Choquet integral based on the traditional λ-measure and the five different SVM predictions with each data set. The right one is the comparison of the Choquet integral based on our improved λ-measure and the five different SVM predictions with each data set. The different colors mean different data sets.

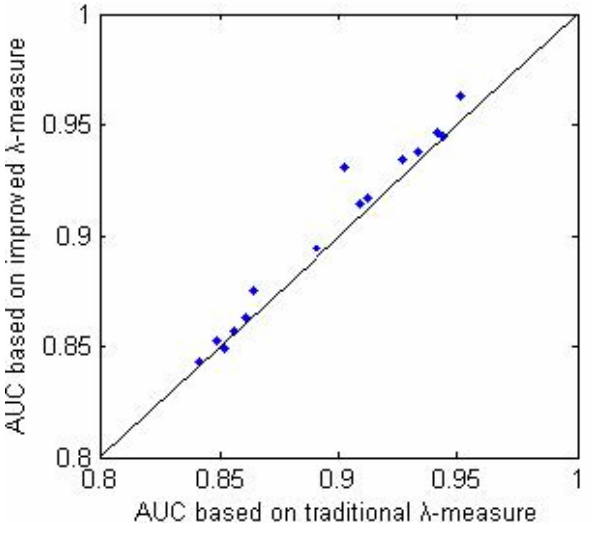


Fig. 2. Scatter-plot of AUCs after versus before our improved Sugeno λ-measure. Points above the diagonal correspond to accuracy improvements by our method. The points represent the AUC of the classes.

It can be seen in Fig. 2 that our improved approach increases AUC for 14 of the 15 classes, comparing with the method based on the traditional Sugeno λ-measure. Just only the AUC of class "32" is less than the latter. It can be seen that the performance of our improved approach is better from the overall perspective.

We also calculate the variances of the fuzzy density determined by the PSO in the two fusion methods' 10-fold cross-validation (shown in Table 2). With the limitation of the space, we just list three classes. We find our improved method can achieve more stable values of fuzzy density using an improved fuzzy measure. This maybe one reason our improved method can get a better result.

Table 2. The variances of the fuzzy density determined by the PSO in the two fusion methods' 10-fold cross-validation

method	class	HMM-logE	Blast	STRING	Exp	Domain
Traditional Choquet	01	0.001651	0.00513	0.002102	0.005376	0.006567
	02	0.009529	0.03821	0.069212	0.003016	0.011522
	10	0.000653	0.00193	0.001528	0.001796	0.001526
Improved Choquet	01	0.000133	9.13e-05	2.4e-05	0.000108	4.69e-05
	02	0.000201	0.00163	0.018385	0.008513	0.000256
	10	0.000371	0.00085	0.001376	0.000269	0.000571

4 Conclusions

In this paper, we use Choquet integral methods to integrate five different data sets. The results of our experiment show that the fusion method can significantly improve the performance of the prediction.

The key of the Choquet integral fusion method is the determination of the fuzzy density. We use PSO to resolve this problem. The typical fitness function may make the particles extremely close to zero, so AUC as the fitness function can avoid the problem and achieve a better result. We also find that scaling the data can get a better result and reduce the computing time when the SVM is used. How to calculate the fuzzy measure in a more feasible way in predicting the protein functions will be our further research.

Acknowledgements. This work was supported by the Science and Technology Development Plan Project of Jilin Province (No. 20070703, No. 20090501).

References

1. Valentini, G.: True Path Rule Hierarchical Ensembles. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 232–241. Springer, Heidelberg (2009)
2. Guan, Y., Myers, C.L., Hess, D.C., Barutcuoglu, Z., Caudy, A.A., Troyanskaya, O.G.: Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biol. 9(Suppl 1), S3 (2008)
3. Barutcuoglu, Z., Schapire, R., Troyanskaya, O.: Hierarchical multi-label prediction of gene function. Bioinformatics 22(7), 830–836 (2006)

4. Xiong, J., Rayner, S., Luo, K., Li, Y., Chen, S.: Genome wide prediction of protein function via a generic knowledge discovery approach based on evidence integration. BMC Bioinformatics 7, 268 (2006)
5. Deng, M., Chen, T., Sun, F.: An integrated probabilistic model for functional prediction of proteins. J. Comput. Biol. 11, 463–475 (2004)
6. Lanckriet, G., Deng, M., Cristianini, N., Jordan, M.I., Noble, W.S.: Kernel-based data fusion and its application to protein function prediction in yeast. Pac. Symp. Biocomput. 9, 300–311 (2004)
7. Chua, H., Sung, W., Wong, L.: An efficient strategy for extensive integration of diverse biological data for protein function prediction. Bioinformatics 23(24), 3364–3373 (2007)
8. Karaoz, U., Murali, T.M., Letovsky, S., Zheng, Y., Ding, C., Cantor, C.R., Kasif, S.: Whole genome annotation using evidence integration in functional linkage networks. Proc. Natl. Acad. Sci. USA 101, 2888–2893 (2004)
9. Chen, Y., Dong, X.: Global protein function annotation through mining genome-scale data in yeast Saccharomyces cerevisiae. Nucleic Acids Res. 32, 6414–6424 (2004)
10. Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., Mewes, H.W.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Res. 32, 5539–5545 (2004)
11. Ashburner, M., Ball, C.A., et al.: Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat. Genet. 25, 25–29 (2000)
12. Sugeno, M.: Theory of fuzzy integrals and its applications. PhD thesis, Tokyo Institute of Technology, Tokyo, Japan (1974)
13. Choquet, G.: Theory of capacities. Annales de l'Institut Fourier 5, 131–295 (1953)
14. Cho, S.B., Jin, H.K.: Multiple network fusion using fuzzy logic. IEEE Transactions on Neural Networks 6(2), 497–501 (1995)
15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 4th IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Service Center, Australia (1995)
16. Shi, Y.H., Eberhart, R.C.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation, pp. 69–73. IEEE Press, Anchorage (1998)
17. John, A.S.: Signal detection theory and ROC analysis in psychology and diagnostics: collected papers. Lawrence Erlbaum Associates, Mahwah (1996)
18. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27, 861–874 (2006)
19. Robert, D., Jaina, M., John, T., Penny, C., Andreas, H., Joanne, E., Pollington, O., et al.: The Pfam protein families database. Nucleic Acids Res. 38, 211–222 (2010)
20. Eddy, S.R.: Profile hidden markov models. Bioinformatics 14(9), 755–763 (1998)
21. Davey, N.E., Shields, D.C., Edwards, R.J.: SLiMDisc: short, linear motif discovery, correcting for common evolutionary descent. Nucleic Acids Res. 34(12), 3546–3554 (2006)
22. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Res. 25, 3389–3402 (1997)
23. Jensen, L.J., Kuhn, M., Stark, M., Chaffron, S., Creevey, C., et al.: STRING 8- a global view on proteins and their functional interactions in 630 organisms. Nucleic Acids Res. 37, 412–416 (2009)

24. Spellman, P., et al.: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomices cerevisiae by microarray hybridization. Mol. Biol. Cell 9, 3273–3297 (1998)
25. Gasch, P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell 11, 4241–4257 (2000)
26. MIPS Comprehensive Yeast Genome Database (CYGD) (2011), http://mips.helmholtz-muenchen.de/genre/proj/yeast/
27. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software http://www.csie.ntu.edu.tw/~cjlin/libsvm
28. Birge, B.: PSOt, A Particle Swarm Optimization Toolbox for Matlab. In: Proceedings of IEEE Swarm Intelligence Symposium, pp. 24–26 (2003)

Gene Clustering Using Particle Swarm Optimizer Based Memetic Algorithm*

Zhen Ji**, Wenmin Liu, and Zexuan Zhu

Shenzhen City Key Laboratory of Embedded System Design,
College of Computer Science and Software Engineering,
Shenzhen University, Shenzhen, 518060, China

Abstract. K-means is one of the most commonly used clustering methods for analyzing gene expression data, where it is sensitive to the choice of initial clustering centroids and tends to be trapped in local optima. To overcome these problems, a memetic K-means (MKMA) algorithm, which is a hybridization of particle swarm optimizer (PSO) based memetic algorithm (MA) and K-means, is proposed in this paper. In particular, the PSO based MA is used to minimize the within-cluster sum of squares and the K-means is used to iteratively fine-tune the locations of the centers. The experimental results on two gene expression datasets indicate that MKMA is capable of obtaining more compact clusters than K-means, Fuzzy K-means, and the other PSO based K-means namely PK-means. MKMA is also demonstrated to attain faster convergence rate and more robustness against the random choice of initial centroids.

1 Introduction

DNA microarray is a high-throughput technology that gives biologists a large amount of gene expression data for studying the cellular gene expression patterns [1]. Gene expression data is normally arranged as a matrix $E = [e_{ij}]$ of size $G \times C$, where G is the number of genes and C denotes the number of experimental conditions. An element e_{ij} represents the corresponding expression level of the i-th gene at the j-th experimental condition. One of the main tasks of microarray data analysis is to identify the co-expressed genes that are likely to have similar biological functions or involve in the same biological process. Clustering techniques are commonly used to fulfil the task by grouping the co-expressed genes into the same cluster.

There have existed numerous clustering methods for the analysis of the gene expression data. Hierarchical clustering [2], one of the earliest cluster methods applied on gene expression data, organizes gene in a hierarchical tree using either bottom-up or top-down approach. In bottom-up approach, hierarchical clustering successively merges pairs of clusters together into a single cluster. On the contrary, the top-down approach starts from

* This work is sponsored by National Natural Science Foundation of China (60872125 & 61001185), Program for New Century Excellent Talents in University, Fok Ying Tung Education Foundation, Guangdong Natural Science Foundation (10151806001000002) and Shenzhen City Funds for Distinguished Young Scientists.

** Corresponding author. Tel.: +86 755 26557413, jizhen@szu.edu.cn

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 587–594, 2011.
© Springer-Verlag Berlin Heidelberg 2011

the whole set as a single cluster and successively splits it into small clusters. Hierarchical clustering has achieved promising results on gene clustering yet its hierarchy is greatly affected by minor perturbations of the input data. Self-organizing map (SOM) [3], a kind of artificial neural network, is another widely used clustering algorithm for gene clustering. SOM is a unsupervised learning method based on a mapping from high-dimensional data to a two-dimensional representation space. It is robust against minor changes on the input data. However, it has trouble with identifying clear-cut clusters. K-means [4], one of the simplest partition clustering methods, categories genes into k groups based on inter-gene similarity/distance. The clustering proceeds to minimize the sum of squared distance between genes and the corresponding cluster centroid. K-means tends to obtain better performance and robustness than hierarchical clustering and SOM on gene expression data [5]. However, the performance of K-means is sensitive to the choice of initial clustering centroids and it could easily result in local minima.

In this paper, we propose a particle swarm optimizer (PSO) [9] based memetic algorithm (MA) [10] to solve the optimization problem involved in K-means, i.e., the minimization of the sum of squared distances. The new algorithms is named as memetic K-means algorithm (MKMA). MA is a newly proposed synergy of evolutionary population-based global search approaches with local refinement heuristic. It always converges to high-quality solutions more efficiently than its conventional counterpart thanks to the advantage of using both global search and local search. MKMA evolves a particle swarm of candidate clustering solutions rather than concerning only one solution in global search to reduce the dependency of the performance on the initial centroids and avoid being trapped in local optimal. A local search is applied on each particle to refine the solution and accelerate convergence in local region. The experimental results on two gene expression datasets indicate that MKMA is capable of handling high-dimensional multi-modal optimization problems efficiently.

The remainder of this paper is organized as follows. Section 2 describes the details of the proposed MKMA. Section 3 presents the experimental design and Section 4 shows the experimental results of the new algorithm on two gene expressing datasets. Finally, Section 5 concludes this study with a short discussion.

2 Methods

MA is a framework of combining global search and local search to take the advantage of both. The main inspiration of MA is Dawkins' notion of "meme", which defines a unit of cultural evolution and represents the evolution law of human beings. In this study, a MA based K-means clustering method is proposed to accelerate the search convergence of the cluster centroids and increase the possibility of identifying global optima. The procedure of MKMA is outlined in Algorithm 1.

2.1 Particle Encoding and Fitness Calculation

In MKMA, each particle is designed to encode candidate positions of K cluster centroids. Each particle is presented as a vector of $K \times L$ elements, where L denotes the dimension of a gene. A shortened K-means is performed using the initial centroid

Algorithm 1. The procedure of MKMA

1. Initialize the position and velocity of each particle in the swarm;
2. Update the velocity and position according to Eq.(2);
3. Evaluate the fitness of the particles;
4. Identify the leader and populace particles;
5. Perform local search on the leader particles;
6. Update on the populace particles using three updating strategies;
7. Evaluate the fitness of the particles;
8. Go to step 2 if the stopping criterion is not satisfied.

positions encoded in each particle. The centroids are refined with only three updating iterations as follows:

Algorithm 2. A shortened K-means

1. Assign each gene to the closest centroid encoded in the particle;
2. Update the centroid by calculating the new means of the genes in the cluster;
3. Repeat steps 1 and 2 for three times.

More iterations could be applied, whereas our empirical studies show that the performance of MKMA does not improve too much with more than three iterations. Therefore, the number of the iterations is set to three in this study to avoid wasting computational efforts. Once the shortened K-means outlined in Algorithm 2 is finished, the fitness of the renewed particle can be calculated based on the mean square error (MSE):

$$\tilde{D} = \frac{1}{M}\sum_{i=1}^{M}[d_{\min}(x_i)]^2 \tag{1}$$

where $d_{\min}(x_i)$ denotes the Euclidean distance between genes x_i to its closest centroid and M is the total number of genes.

2.2 Global Search

The global search in MKMA is conducted with the comprehensive learning particle swarm optimizer (CLPSO) [11]. Using a novel learning strategy whereby all other particles' historical best information is used to update a particle's velocity, CLPSO maintains the diversity of the swarm to improve the exploration efficiency of global search. The velocity update equations of CLPSO are defined as:

$$\begin{aligned} V_i^{k+1} &= \omega V_i^k + cr^k(pbest(f_i)^k - P_i^k) \\ P_i^{k+1} &= P_i^k + V_i^{k+1} \end{aligned} \tag{2}$$

where V_i^{k+1} and P_i^{k+1} denote the i-th particle's velocity and position in iteration $k+1$, respectively. Variable ω is an additional inertia weight for balancing the exploration and exploitation on the search space, c is a constant learning factor, r is a random number

in [0,1], and $pbest(f_i)$ represents the the best previous position of particle f_i where f_i^d is set to i in a learning probability of P_c, otherwise, two particles are randomly selected from the swarm using tournament selection and f_i is assigned the one with better fitness.

2.3 Local Search

In each iteration of CLPSO, the particle swarm is partitioned into a leader and a populace group based on fitness value.

One third of the best particles in the swarm are considered as leader particles. The remainder particles in the swarm form the populace group. The leader particles are assumed to be closer to the optimal solutions and they are worth spending local refinement to reach the optima. For the conventional PSO [9] has strong ability of exploitation in local region, the local search of MKMA fine-tunes the leader particles based on the updating rules of PSO:

$$\begin{aligned} V_i^{k+1} &= \omega V_i^k + c_1 r_1^k (pbest_i^k - P_i^k) + c_2 r_2^k (gbest^k - P_i^k) \\ P_i^{k+1} &= P_i^k + V_i^{k+1} \end{aligned} \tag{3}$$

where c_1 and c_2 are the learning factors, r_1 and r_2 are the random values $[0, 1]$, $pbest_i$ represents the previous best position of particle i, and $gbest$ denotes the best position of all particles.

For the populace particles, they each adopts one of the following three strategies based on the performance improvement of $gbest$.

- **Approaching strategy:** In this strategy, particle is moved toward $gbest$ based on the updating equation:

$$P_i^{k+1} = P_i^k + r(gbest^k - P_i^k) \tag{4}$$

- **Random strategy:** In this strategy, particle updates its position randomly to increase the diversity of swarm. The updating equation is defined as follows:

$$P_i^{k+1} = rand(P_{\min}, P_{\max}) \tag{5}$$

  where $[P_{min}, P_{max}]$ is the range of particle's position.
- **Dispersal strategy:** By performing this strategy, the particles push away $gbest$ according to the equation:

$$P_i^{k+1} = P_i^k - r(gbest^k - P_i^k) \tag{6}$$

Let the performance improvement of $gbest$ be δ:

$$\delta = \frac{(lastvalue - fitness(gbest))}{lastvalue} \tag{7}$$

where $lastvalue$ denotes the previous fitness value of $gbest$. The fitness value $fitness$ $(gbest)$ is considered as decreasing fast if $\delta \geq 0.01$. In that case $gbest$ is probably

located in a promising region that is worth more exploration, hence populace particles will adopt approaching strategy and move toward $gbest$. If $\delta < 0.001$, $fitness(gbest)$ is stagnant, which suggests $gbest$ could be trapped in a local optima, therefore the dispersal strategy is applied to keep the populace particles away from the local optima and search a larger region in the solution space. Otherwise, when $\delta \in [0.001, 0.01)$, the random strategy is used to increase the diversity of the swarm.

3 Experimental Design

To evaluate the performance of MKMA, comparison study to other K-means based algorithms including the conventional K-means, Fuzzy K-Means (FKM) [6] and PK-means [7] is conducted on two gene expression datasets.

FKM extends K-means by assigning each point to more than one clusters in terms of the degree of membership. FKM is less sensitive to the initial partitions but it is much more time-consuming than K-means. PK-means is a newly proposed PSO based K-means for gene clustering. It bridges particle-pair optimizer [8] and K-means to outperform K-means and FKM in terms of convergence rate and computational time.

Two datasets are used to evaluate the performance of the clustering algorithms:

- Yeast cell-cycle data [12] consists of 6179 genes and 77 dimensions. Genes missing more than 20% were deleted and 5571 genes are obtained after processing and filtering. For the genes retained for analysis, missing values were imputed by KNN algorithm [13].
- Sporulation data [14] contains 6120 genes with 7 dimensions. It assays nearly every yeast gene changes in gene expression during sporulation. After processing and filtering procedure, there are 6039 genes retained for analysis.

The number of clusters K for all clustering algorithms is set to 256 on each dataset. The parameters of PK-means are set according to [7]. For K-means, FKM, and MKMA the iterations are terminated if the following condition holds.

$$\frac{(\tilde{D}^{(k-1)} - \tilde{D}^{(k)})}{\tilde{D}^{(k)}} \leq \varepsilon \tag{8}$$

where $\tilde{D^k}$ is the mean square error (defined in Eq. 1) in iteration k. The key parameter values used in K-means, FKM, PK-means and MKMA are listed in Table 1. The other parameters of CLPSO are set according to [11].

4 Results

4.1 Mean Square Error, Homogeneity and Separation

In addition to MSE $\tilde{D}$, the other two indices, homogeneity(D_1) and separation(D_2), are also introduced for measuring the clustering performance.

Homogeneity calculates the average distance between each gene expression profile and its closest controid as follows:

$$D_1 = \frac{1}{M} \sum_i d(x_i, c(x_i)) \tag{9}$$

Table 1. Parameter values

Algorithms	ε	λ	ω	c	c_1	c_2
K-means	0.001	–	–	–	–	–
FKM	0.001	10	–	–	–	–
PK-means	–	–	0.1	–	0.3	0.5
MKMA	0.001	–	0.1	0.3	0.3	0.5

where x_i is the i-th gene, $c(x_i)$ is the i-th center of the cluster, and d is the Euclidean distance function. Separation is calculated as the weighted average distance between cluster centers:

$$D_2 = \frac{1}{\sum_{i \neq j} N_{C_i} N_{C_j}} \sum_{i \neq j} N_{C_i} N_{C_j} d(C_i, C_j) \tag{10}$$

where C_i and C_j represent the centers of the i-th and j-th clusters, respectively. N_{C_i} and N_{C_j} denote the number of genes in the i-th and j-th clusters, respectively.

D_1 reflects the compactness of the clusters while D_2 reflects the overall distance between clusters. Decreasing D_1 and $\tilde{D}$ or increasing D_2 suggests an improvement in the clustering results.

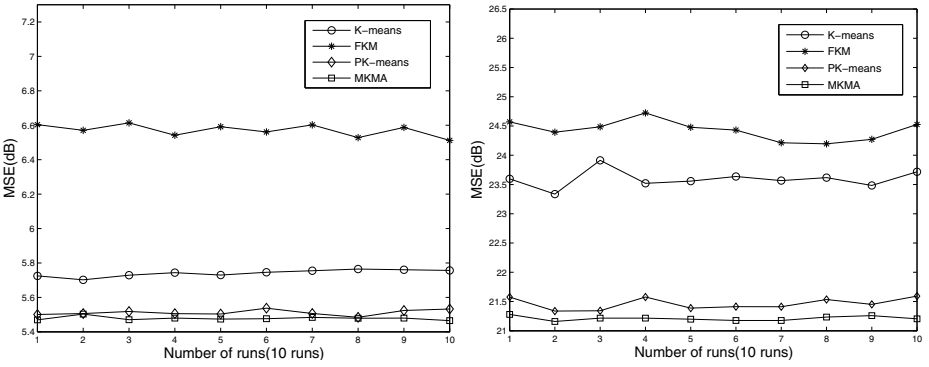
Table 2. MSE, homogeneity and separation

Dataset	Algorithm	$\tilde{D}$	D_1	D_2
Yeast cell-cycle	K-means	5.73	2.30	3.16
	FKM	6.58	2.44	2.77
	PK-means	5.52	2.29	3.20
	MKMA	**5.48**	**2.28**	**3.22**
Lymphoma	K-means	23.38	4.59	7.49
	FKM	24.39	4.66	7.11
	PK-means	21.44	4.47	7.62
	MKMA	**21.21**	**4.45**	**7.66**

The average MSE, homogeneity and separation values over 10 random runs of the four comparing clustering algorithms are summarized in Table 2. It is shown that MKMA attains competitive or better performance than K-means, FKM, and PK-means in terms of inter- and intra-cluster distance.

4.2 Sensitivity to the Choice of Initial Centroids

To evaluate the sensitive of the final clustering performance to the choice of initial centroids. The four algorithms are randomly run 10 times with different initial centroids. The results shown in Fig.1 indicate that the performance of both PK-means and MKMA is less sensitive to different selections of initial clustering centroids than K-means and FKM. MKMA is observed to obtain smaller MSN than other three clustering methods.

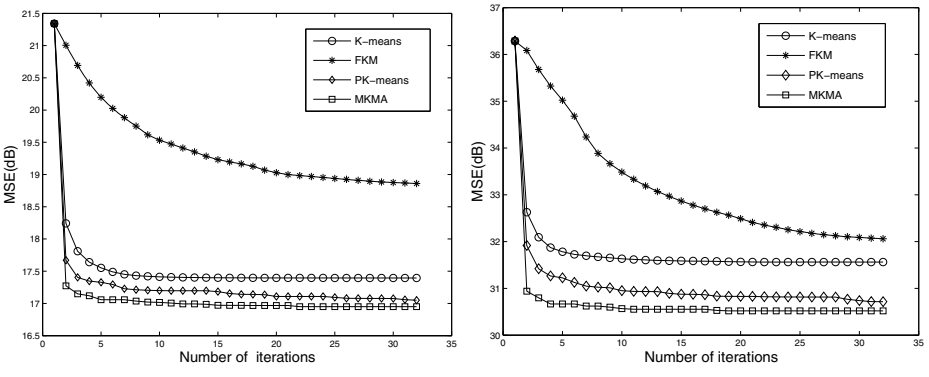


(a) MSE plots on yeast cell-cycle data (b) MSE plots on lymphoma data

Fig. 1. MSE plots by 10 randomly runs on (a) yeast cell-cycle data (b) lymphoma data

4.3 Convergence Rate

The convergence trace of all algorithms is depicted in Fig.2, where MSE value is plotted against the number of iterations on updating the centroid positions. It can be seen that MKMA converges faster to better MSE values than the other three clustering algorithms on the two datasets.



(a) MSE plots on yeast cell-cycle data (b) MSE plots on lymphoma data

Fig. 2. MSE plots by iterations on (a) yeast cell-cycle data (b) lymphoma data

5 Conclusion and Discussion

This paper presented a novel memetic K-means algorithm (MKMA) for gene expression data. MKMA applies particle swarm optimized based memetic algorithm to solve the optimization problem of K-means. The experimental results on two gene expression datasets suggest that MKMA consistently attain competitive or better performance than K-means, FKM, and PK-means. MKMA is demonstrated to be less sensitive to the initial clustering centroids, and it has fast convergence rate.

References

1. Schena, M., Shalon, D., Davis, R.W., et al.: Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. Science 270(5235), 467–470 (1995)
2. Milligan, G.W., Cooper, M.C.: A Study of the Comparability of External Criteria for Hierarchical Cluster Analysis. Multivariate Behavioral Research 21, 441–458 (1986)
3. Fritzke, B.: Growing Cell Structures-a Self-organizing Network for Unsupervised and Supervised Learning. Networks 7, 1141–1160 (1994)
4. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 271–297. University of California Press, Berkeley (1967)
5. Thalamuthu, A., Mukhopadhyay, I., Tseng, G.C.: Evaluation and Comparison of Gene Clustering Methods in Microarray Analysis. Bioinformatics 22(8), 2405–2412 (2006)
6. Gasch, A.P., Eisen, M.B.: Exploring the Conditional Coregulation of Yeast Gene Expression through Fuzzy K-means Clustering. Genome Biology 3(11), 1–22 (2002)
7. Du, Z.H., Wang, Y.W., Ji, Z.: PK-means:a New Algorithm for Gene Clustering. Computatoinal Biology and Chemistry 32, 243–247 (2008)
8. Ji, Z., Liao, H.L., Xu, W., Jiang, L.: A Strategy of Particle-pair for Vector Quantization in Image Coding. Acta Electron. Sin. 35(7), 86–89 (2007)
9. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Network, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
10. Moscato, P.: Memetic Algorithm: a Short Introduction. McGraw-Hill, London (1999)
11. Liang, J.J., Qin, A.K., et al.: Comprehensive Learing Particle Swarm Optimizer for Global Optimization of Multimodal Functions. IEEE Transactions on Evolutionary Computation 10(3), 281–295 (2006)
12. Spellman, P.T., et al.: Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces Cerevisine by Microarray Hybridization. Mol. Biol. Cell 9, 3273–3297 (1998)
13. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Bostein, D., Altman, R.B.: Missing Value Estimation Methods for DNA Microarrays. Bioinformatics 17, 520–525 (2001)
14. Chu, S., et al.: The Transcriptional Program of Sporulation in Budding Yeast. Science 282, 699–705 (1998)
15. Alizadeh, A.A., et al.: Distinct Types of Diffuse Large B-cell Lymphoma Identified by Gene Expression Profiling. Nature 403, 503–511 (2000)

Hybrid Particle Swarm Optimization with Biased Mutation Applied to Load Flow Computation in Electrical Power Systems

Camila Paes Salomon, Maurilio Pereira Coutinho,
Germano Lambert-Torres, and Cláudio Ferreira

Federal University of Itajuba, Av. BPS, 1303, Pinheirinho,
37.500-503, Itajuba, MG, Brazil
{camsalomon,maurilio.coutinho,germanoltorres}@gmail.com,
claudiof@unifei.edu.br

Abstract. This paper presents the implementation of a Hybrid Particle Swarm Optimization with Biased Mutation (HPSOBM) algorithm to solve the load flow computation in electrical power systems. The load flow study obtains the system status in the steady-state and it is widely used in the power system operation, planning and control. The proposed methodology is applied in a different computational model, which is based on the minimization of the power mismatches in the system buses. This new model searches for a greater convergence, and also a larger application in comparison with traditional numerical methods. In order to illustrate the proposed algorithm some simulations were conducted using the IEEE 14 bus system.

Keywords: Particle Swarm Optimization, Load Flow, Evolutionary Computation, Artificial Intelligence, Electrical Power Systems.

1 Introduction

Modern operation centers have managed electrical power systems remotely and automatically, accomplishing functions such as automatic generation control, state estimation, topology analysis, etc [1]. The load flow studies are required by most of these functions. Load flow is an electrical engineering known problem which determines the power system operation point in the steady-state [2]. The load flow – or power flow – algorithm calculates the buses voltages and the amount of power in the system generation buses as well as the power flow in the system branches. A set of non-linear equations is applied to model this kind of problem, which is commonly solved using numerical methods [3]. Among the traditional numerical methods used for load flow computation the Newton-Raphson approach has a better and a faster convergence. However, such method has some difficulties because of the complex Jacobian matrix calculation and inversion and also the dependence on good initial estimated values to guarantee the convergence. Moreover, some present changes in the power system characteristics, such as an occurrence of a higher R/X ratio, may

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 595–605, 2011.
© Springer-Verlag Berlin Heidelberg 2011

complicate the load flow convergence [6]. Thus these difficulties have been inducing the researches to concentrate efforts in order to develop alternative methods to solve the load flow equations. These new methods look for an easier and a more efficient implementation and less computational time, as well as overcoming some possible limitations and convergence problems. Many of these new methods have applied artificial intelligence techniques.

Many researchers in the area of artificial intelligence have been putting their attentions in the computational intelligence algorithms, and Particle Swarm Optimization (PSO) is pointed out among these techniques. PSO algorithms are applied in function optimization and they are based on the behavior of birds' flocks searching for food [10]. PSO has been applied in several themes related to electrical power systems and they have provided good convergence properties, ease of implementation and good computational time [5]. In [7] the author approaches the PSO to the load flow calculation in shipboard systems. In [5,8,14] the authors propose the PSO application to the optimal power flow (OPF) problem. In [11] the author proposes a PSO methodology applied to power system restoration. In [12] it is presented the PSO applied to voltage and reactive power control. Besides PSO application, many researches have been implementing hybrid models, joining PSO with other techniques, for instance the Genetic Algorithm (GA) operators. In [4] it is proposed a Hybrid PSO with Mutation applied to loss power minimization, and in [6] it is presented a chaotic PSO algorithm with local search to the load flow calculation, overcoming some limitations found in the traditional methods.

The paper proposes the application of a Hybrid Particle Swarm Optimization with Biased Mutation (HPSOBM) algorithm to the load flow computation. This methodology is based on the minimization of the apparent power mismatches in the system buses. The involved variables are continuous and must remain within the specified boundaries in the system input data. Experiments for the proposed methodology were accomplished using the IEEE 14-bus system using the software implemented.

2 Power System Load Flow Analysis

The power flow study provides the system status in the steady-state; it consists in the determination of the possible power system operational states through the previous knowledge of some variables of the system buses. This study aims to obtain the system buses voltages in order to determine later the power adjustments in the generation buses and the power flow in the system branches [3]. After the system steady-state is calculated, it is possible to obtain the amount of power generation necessary to supply the power demand plus the power losses in the system branches. Besides, the voltage levels must remain within the boundaries and overloaded operations added to those in the stability limit must be prevented [3]. The general form of the Static Load Flow Equations (SLFE) is given by (1):

$$P_i - jQ_i - y_{i1}V_1V_i^* - y_{i2}V_2V_i^* - \ldots - y_{in}V_nV_i^* = 0 \quad . \tag{1}$$

where: i = 1,..., n, bus number; P_i = active power generated or injected in the bus i; Q_i = reactive power generated or injected in the bus i; $|V_i|$ = voltage module of the bus i; δ_i = voltage angle of the bus i; $V_i = |V_i|e^{j\delta i}$, i. e., the voltage in the polar form; $V_i^* = |V_i|e^{-j\delta i}$, i. e., the conjugate voltage; y_{ik} = element of the nodal admittance matrix Y_{bus}.

The nodal admittance matrix can be computed as follows: if i = k, y_{ik} is the sum of the admittances that come out from the bus i; else y_{ik} is the admittance between the buses i and k, multiplied by -1.

The power system buses are classified according to the variables previously known and to the variables that will be calculated later through the SLFE. Type 1 Bus or PQ Bus: P_i and Q_i are specified and $|V_i|$ and δ_i are calculated; Type 2 Bus or PV Bus: P_i and $|V_i|$ are specified and Q_i and δ_i are calculated; Type 3 Bus or Vδ Bus ("Slack Bus"): $|V_i|$ and δ_i are specified and P_i and Q_i are calculated.

A complex and non-linear equations system is represented by (1), so its solution is obtained through approximations using numeric methods. These methods make the assumption of the initial estimate values to the bus voltages and in the application of the SLFE in successive iterations, looking for better approximations. The required accuracy determines the stop criterion.

3 Hybrid Particle Swarm Optimization with Biased Mutation Applied to Load Flow Computation

Swarm Intelligence is a kind of Artificial Intelligence based on social behavior, i. e., the behavior of the animals living in groups and having some ability to interact among each other as well as with the environment in which they are inserted [10,15]. PSO is an optimization algorithm developed through the simulation of simplified social models as bird flocks flying randomly in search for food [10,15].

PSO is applied to function optimization and it uses a population of individuals, i. e., a set of particles, where each particle is a candidate to the solution of the problem. These particles are distributed in the search space, each one having a determined position and velocity at each time instant. Moreover, such particles have knowledge about their performances and also about their neighbors' performances. The best individual position of a particle is called *personal best*, and the best position of all the particles is called *global best*. The *rule function* is the evaluation function which performs the interaction between the particles and the environment in which they are inserted, and it is related to the problem modeling [10,15].

The PSO algorithm analyzes, at each time step, the displacement of each particle in search for the best position and updating its velocity and position through defined equations. This process is iterative and it proceeds until all the particles converge to the achieved global best, which is adopted as the problem solution.

3.1 Mutation Operation in Genetic Algorithms

Genetic Algorithm (GA) is a search heuristic iterative procedure that uses a population of individuals, in which each one represents a possible solution to the treated

problem [16,17]. It is based on Darwin's natural selection, inspired by the evolution process [18]. The GA accumulated information is applied to decrease the search space and create new good solutions to the problem domain [19].

The GA mutation operation is used in order to cover better the state space and also to prevent the GA from converging to a local best. The mutation is performed by mutating the individual gene value in a random way with a determined probability, i. e., some individuals of the new population may have one of their genes – or nature – randomly modified [19].

3.2 Definition of the Proposed Algorithm

The presented HPSOBM (Hybrid Particle Swarm Optimization with Biased Mutation) algorithm is based on the minimization of the power mismatches in the system buses. The methodology consists in the adoption of initial estimated values for the particles positions, which are defined as the buses voltages, and updating them at each process' iteration using the HPSOBM equations. Once particles are defined as the buses voltages, they assume continuing values within the boundaries specified in the input data. The rule function parameters are defined as *grades* and they must be minimized in the HPSOBM algorithm. The grades are computed as the arithmetic mean of the buses apparent power. Each particle has a *personal grade*, i. e., the value obtained by its personal best. The *global grade* is the grade associated to the global best. The *current grade* is the grade obtained by a particle at the current iteration of the process.

The algorithm begins generating the initial estimate value to the position of the particles, velocities, personal best values and global best values. The voltage angle begins with a random initial value within the specified boundary. In the case of a PQ bus, the voltage module begins with a random value within the specified boundary; for a PV bus, the voltage module receives the related value specified in the input data. The initial velocities are null. The personal best parameters start with the position of the particle values and the global best parameter starts with an arbitrary particle value. The grades begin with high values in order to be minimized later. Thus the iterations are initialized. The procedure explained as follows is executed for each particle of the population, which was fixed in 15 particles. Firstly the buses voltages start with the position of the particle. Thus the reactive power of the PV buses is computed using equation (1), then the active and reactive power of the Vδ bus are also computed using (1). The power flow in the system branches is calculated using the equation (2).

$$S_{ij} = P_{ij} + jQ_{ij} = V_i(V_i^* - V_j^*)Y_{ij}^* + V_i V_i^* Y_{sh,i} \quad . \tag{2}$$

where: S_{ij} = complex apparent power between the buses i and j; P_{ij} = active power between the buses i and j; Q_{ij} = reactive power between the buses i and j; V_i = bus i voltage; V_j = bus j voltage; $V_i^* = |V_i|e^{-j\delta i}$, i. e., the conjugate voltage; $V_j^* = |V_j|e^{-j\delta j}$, i. e., the conjugate voltage; Y_{ij} = admittance between the buses i and j; $Y_{sh,i}$ = shunt admittance of the bus i.

The active and reactive power mismatches of each bus are calculated as the sum of the injected power in the approached bus and the apparent power mismatches arithmetic mean is obtained. It is also obtained the particle position which has the worst – the biggest – power mismatch until now. This particle index is kept and it is used in the mutation operation. The personal best updating verification is made. After all the particles pass through the described routine, it is made the global best updating verification. The velocities as well as the position of the particles are updated according to the equations (3), (4) and (5); which are, respectively: velocities equation, positions equation and inertia weight equation, applied to each particle [4,9-11].

$$v(t+1) = w.v(t) + c_1.r_1.(p(t) - x(t)) + c_2.r_2.(g(t) - x(t)) \quad . \tag{3}$$

$$x(t+1) = x(t) + v(t+1) \quad . \tag{4}$$

$$w = w_{\max} - \frac{(w_{\max} - w_{\min}).t}{ni} \quad . \tag{5}$$

where: i = particle index; t = iterations counter; ni = total number of iterations; $v(t)$ = particle i velocity at iteration t; $x(t)$ = particle i position at iteration t; r_1, r_2 = random numbers between 0 and 1; c_1, c_2 = acceleration coefficients, both set to a value of 2.0; $p(t)$ = particle i personal best found at iteration t; $g(t)$ = global best found at iteration t; w = velocity equation's inertia weight, w_{max} = inertia weight maximum value, set to a value of 0.7; w_{min} = inertia weight minimum value, set to a value of 0.2.

Then, the mutation operation is applied. This operation aims to coverage better the problem domain and to obtain a new particle, avoiding a premature convergence to a local best point. The mutation is applied to the worst particle of the current iteration, i. e., the particle which has the bigger power mismatch value, and because of this philosophy it is called Biased Mutation. The procedure consists in adding a random value to the particle voltage module and angle, according to (6).

$$mx(k) = x(k) + 0.1.[(x_{\max} - x_{\min}).r + x_{\min}] \quad . \tag{6}$$

where: k = mutated particle index; $x(k)$ = particle position before the mutation operation; $mx(k)$ = particle position after the mutation operation; r = random number between 0 and 1; x_{max} = maximum value of the position, related to the specified boundary in the input data; x_{min} = minimum value of the position, related to the specified boundary in the input data.

Finally, in the end of the iterations, it is obtained the final global best, which is adopted as the load flow solution. It is important to notice that the HPSOBM methodology can achieve several acceptable results for the same load flow study, depending on the simulation. It occurs because each particle has a random initial estimate value and the HPSOBM equations also make use of random values, so several solutions can

be achieved for the same initial estimative. However, numeric traditional methods start with the same initial estimative values and achieve the same final results, regardless of the program simulation.

4 Numerical Results

The results obtained through the proposed HPSOBM algorithm are presented in this section. The IEEE 14 bus test system, Fig. 1, is used for this purpose. The transformers taps were kept in the rated positions. The implemented version of this algorithm obtains the power flow solution based on the minimization of the power mismatches in the system buses. Tables 1 and 2 present the power flow results for the test system, obtained through a conventional program applying the Newton-Raphson method.

Two different and arbitrary simulations were accomplished using the proposed methodology, simulation 1 and simulation 2. Tables 3 and 4 present the results related to the simulation 1 and Tables 5 and 6 are related to the simulation 2. Each simulation has a different solution for the power flow due to the HPSOBM algorithm nature. The solutions are valid because the values are still within the permitted limits.

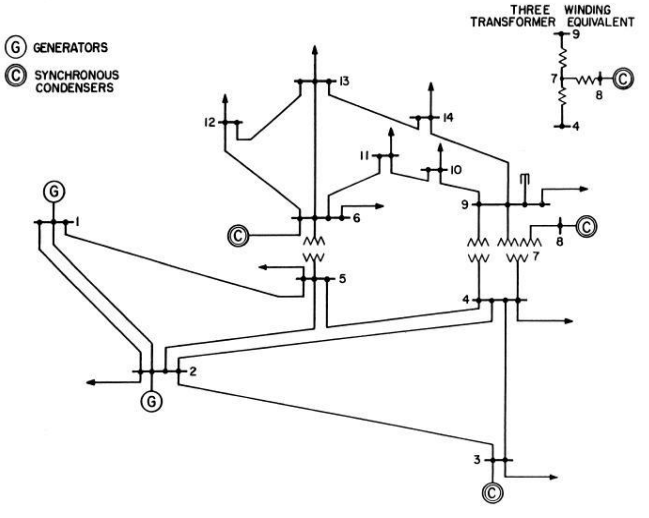


Fig. 1. IEEE 14-bus test system, with all system buses, branches and elements

The odd-numbered tables contain the results related to the system buses. The three last columns of the table represent the active, reactive and apparent power mismatches, respectively. The even-numbered tables represent the results related to the system branches. These tables show the power flow in the system lines computed for the voltage solutions found in each simulation.

Table 1. Simulation using Newton-Raphson method: buses parameters results. k = bus index, V_k = bus voltage module k, δ_k = bus voltage angle k, P_k = active power generated at bus k, Q_k = reactive power generated at bus k, ΔP_k = active power mismatch at bus k, ΔQ_k = reactive power mismatch at bus k, ΔS_k = apparent power mismatch at bus k.

k	V_k	δ_k	P_k	Q_k	ΔP_k	ΔQ_k	ΔS_k
1	1,060000	0,000000	2,304174	-0,222584	-5,551115E-16	1,856154E-15	1,937384E-15
2	1,045000	-0,085521	0,183000	0,169491	-2,771615E-02	-3,025358E-15	2,771615E-02
3	1,010000	-0,219911	-0,942000	0,009071	-4,235448E-03	1,000935E-15	4,235448E-03
4	1,026000	-0,181514	-0,478000	0,039000	6,141135E-02	-2,113986E-03	6,144773E-02
5	1,033000	-0,155334	-0,076000	-0,016000	-4,482804E-02	-6,610994E-03	4,531290E-02
6	1,070000	-0,260054	-0,112000	0,422322	1,697325E-02	4,607426E-15	1,697325E-02
7	1,045000	-0,233874	0,000000	0,000000	-4,328546E-03	-3,045509E-04	4,339247E-03
8	1,090000	-0,233874	0,000000	0,278456	2,954397E-17	1,887379E-15	1,887610E-15
9	1,028000	-0,261799	-0,295000	-0,166000	-2,019721E-02	5,011505E-03	2,080968E-02
10	1,028000	-0,267035	-0,090000	-0,058000	8,420636E-03	-6,329197E-03	1,053403E-02
11	1,045000	-0,265290	-0,035000	-0,018000	-1,657940E-03	2,750084E-03	3,211188E-03
12	1,053000	-0,274017	-0,061000	-0,016000	-2,235718E-03	-1,006758E-04	2,237984E-03
13	1,046000	-0,274017	-0,135000	-0,058000	-7,457740E-03	9,407115E-03	1,200465E-02
14	1,018000	-0,286234	-0,149000	-0,050000	4,933314E-03	-5,509430E-03	7,395364E-03

Table 2. Simulation using Newton-Raphson method: power flow in the system lines. $P_{ij\ (ji)}$ = active power in the line composed by the buses i-j (j-i), $Q_{ij\ (ji)}$ = reactive power in the line composed by the buses i-j (j-i).

i	j	P_{ij}	Q_{ij}	P_{ji}	Q_{ji}
1	2	1,543861	-0,198190	-1,502260	0,266711
1	5	0,760313	-0,024394	-0,732515	0,085256
2	3	0,728546	0,035980	-0,705552	0,014638
2	4	0,567517	-0,064999	-0,550264	0,080890
2	5	0,416914	-0,068200	-0,407722	0,058911
3	4	-0,232213	-0,005567	0,235755	0,001343
4	5	-0,645319	0,042655	0,650624	-0,025924
4	7	0,268329	-0,086193	-0,268329	0,101972
4	9	0,152088	0,002419	-0,152088	0,009805
5	6	0,458441	-0,127633	-0,458441	0,181117
6	11	0,076296	0,098133	-0,075014	-0,095449
6	12	0,077887	0,034114	-0,077111	-0,032499
6	13	0,175285	0,108958	-0,172824	-0,104112
7	8	0,000000	-0,266960	0,000000	0,278456
7	9	0,272658	0,165293	-0,272658	-0,155051
9	10	0,057411	-0,021441	-0,057298	0,021741
9	14	0,092532	-0,004325	-0,091499	0,006520
10	11	-0,041123	-0,073412	0,041672	0,074699
12	13	0,018347	0,016599	-0,018225	-0,016489
13	14	0,063506	0,053193	-0,062434	-0,051011

Table 3. Simulation 1 using the proposed HPSOBM algorithm: buses parameters results

k	V_k	δ_k	P_k	Q_k	ΔP_k	ΔQ_k	ΔS_k
1	1.060000	0.000000	2.325316	-0.225075	-5.551115E-16	1.162265E-15	1.288025E-15
2	1.045000	-0.086509	0.183000	0.180537	-5.203014E-09	2.414735E-15	5.203014E-09
3	1.010000	-0.220486	-0.942000	0.010455	-3.729702E-09	1.306247E-15	3.729702E-09
4	1.026096	-0.180923	-0.478000	0.039000	-3.670963E-08	-4.293257E-08	5.648719E-08
5	1.032600	-0.156152	-0.076000	-0.016000	-1.587668E-06	-2.858496E-06	3.269815E-06
6	1.070000	-0.259695	-0.112000	0.416080	8.960580E-09	-4.996004E-16	8.960580E-09
7	1.044823	-0.234759	0.000000	0.000000	3.180789E-08	-1.617875E-07	1.648847E-07
8	1.090000	-0.234759	0.000000	0.279552	8.644264E-10	2.220446E-15	8.644264E-10
9	1.027653	-0.263027	-0.295000	-0.166000	2.065322E-07	-3.049806E-06	3.056791E-06
10	1.027578	-0.267366	-0.090000	-0.058000	2.813536E-05	-2.468896E-04	2.484876E-04
11	1.044961	-0.265533	-0.035000	-0.018000	1.956981E-08	-3.222618E-08	3.770284E-08
12	1.053019	-0.274361	-0.061000	-0.016000	5.053687E-08	-3.887773E-08	6.376090E-08
13	1.046237	-0.274686	-0.135000	-0.058000	3.074048E-08	-6.885231E-08	7.540304E-08
14	1.017448	-0.286134	-0.149000	-0.050000	6.936667E-07	-3.002547E-06	3.081633E-06

Table 4. Simulation 1 using the proposed HPSOBM algorithm: power flow in the system lines

i	j	P_{ij}	Q_{ij}	P_{ji}	Q_{ji}
1	2	1.560964	-0.202203	-1.518423	0.273593
1	5	0.764352	-0.022872	-0.736257	0.084979
2	3	0.726414	0.036191	-0.703553	0.013869
2	4	0.558508	-0.063507	-0.541802	0.077735
2	5	0.416501	-0.065740	-0.407340	0.056373
3	4	-0.238447	-0.003414	0.242183	-0.000318
4	5	-0.609731	0.042531	0.614468	-0.027590
4	7	0.275863	-0.084462	-0.275863	0.100993
4	9	0.155486	0.003514	-0.155486	0.009264
5	6	0.453131	-0.129759	-0.453131	0.182269
6	11	0.079140	0.097005	-0.077840	-0.094283
6	12	0.080392	0.032875	-0.079582	-0.031190
6	13	0.181599	0.103930	-0.179070	-0.098949
7	8	0.000000	-0.267966	0.000000	0.279552
7	9	0.275863	0.166972	-0.275863	-0.156494
9	10	0.047841	-0.016977	-0.047764	0.017183
9	14	0.088508	-0.001790	-0.087565	0.003796
10	11	-0.042265	-0.074936	0.042840	0.076283
12	13	0.018582	0.015190	-0.018467	-0.015086
13	14	0.062537	0.056035	-0.061436	-0.053793

Table 5. Simulation 2 applying the HPSOBM algorithm: buses parameters results

k	V_k	δ_k	P_k	Q_k	ΔP_k	ΔQ_k	ΔS_k
1	1.060000	0.000000	2.323085	-0.224850	-2.220446E-16	3.132911E-15	3.140769E-15
2	1.045000	-0.086423	0.183000	0.179706	1.146248E-09	-3.330669E-16	1.146248E-09
3	1.010000	-0.220338	-0.942000	0.010169	2.729703E-09	1.042569E-15	2.729703E-09
4	1.026146	-0.180739	-0.478000	0.039000	-7.088421E-04	-1.718341E-04	7.293724E-04
5	1.032639	-0.155997	-0.076000	-0.016000	3.467779E-08	-7.061203E-09	3.538940E-08
6	1.070000	-0.259436	-0.112000	0.415807	3.487077E-07	-2.331468E-15	3.487077E-07
7	1.044870	-0.234460	0.000000	0.000000	-9.817892E-07	-2.709120E-07	1.018481E-06
8	1.090000	-0.234460	0.000000	0.279258	-4.714854E-08	8.881784E-16	4.714854E-08
9	1.027724	-0.262667	-0.295000	-0.166000	-8.835327E-04	-3.555616E-04	9.523938E-04
10	1.027643	-0.266990	-0.090000	-0.058000	-4.628260E-04	-1.307619E-04	4.809434E-04
11	1.044996	-0.265215	-0.035000	-0.018000	-7.659737E-09	-2.820208E-10	7.664927E-09
12	1.053017	-0.274104	-0.061000	-0.016000	7.420782E-05	-4.456680E-06	7.434153E-05
13	1.046248	-0.274417	-0.135000	-0.058000	3.224100E-06	-2.196902E-07	3.231576E-06
14	1.017493	-0.285812	-0.149000	-0.050000	8.714802E-09	-2.711772E-08	2.848366E-08

Table 6. Simulation 2 applying the HPSOBM algorithm: power flow in the system lines

i	j	P_{ij}	Q_{ij}	P_{ji}	Q_{ji}
1	2	1.559486	-0.201857	-1.517027	0.272997
1	5	0.763599	-0.022993	-0.735560	0.084870
2	3	0.726093	0.036223	-0.703251	0.013754
2	4	0.557887	-0.063657	-0.541217	0.077773
2	5	0.416047	-0.065857	-0.406905	0.056430
3	4	-0.238749	-0.003585	0.242494	-0.000125
4	5	-0.609044	0.042557	0.613770	-0.027650
4	7	0.275302	-0.084483	-0.275302	0.100953
4	9	0.155173	0.003450	-0.155173	0.009275
5	6	0.452695	-0.129649	-0.452695	0.182056
6	11	0.078798	0.096978	-0.077503	-0.094265
6	12	0.080403	0.032877	-0.079593	-0.031191
6	13	0.181494	0.103896	-0.178967	-0.098920
7	8	0.000000	-0.267695	0.000000	0.279258
7	9	0.275303	0.166743	-0.275303	-0.156304
9	10	0.047684	-0.016850	-0.047607	0.017055
9	14	0.088676	-0.001765	-0.087730	0.003779
10	11	-0.041930	-0.074924	0.042503	0.076265
12	13	0.018518	0.015196	-0.018404	-0.015092
13	14	0.062368	0.056013	-0.061270	-0.053779

It is possible to test the effectiveness of the proposed methodology analyzing the obtained power mismatches, which must be as small as possible. The voltage modules obtained through the Newton-Raphson based program have accuracy until the third decimal place, so it obscures a direct comparison between the power mismatches obtained through conventional method and through the proposed methodology, because the power mismatches accuracy also depends on the voltage modules accuracy. The HPSOBM proposed algorithm presents voltage modules and angles very close to those found using the Newton-Raphson method and the power mismatches are well

minimized, most of the obtained values are smaller than the tolerance usually accepted, which is about 10^{-4}. Therefore, the proposed methodology efficacy is proved, because it provides results that are better or as good as those obtained using the Newton-Raphson based program.

5 Conclusion

The paper presents a load flow solution methodology based on a Hybrid Particle Swarm Optimization with Biased Mutation (HPSOBM) algorithm. The methodology was tested for the IEEE 14-bus system through the developed computational program. The main advantages of the proposed methodology are the flexibility of implementation and its better convergence.

Analyzing the results for the simulations, it is possible to conclude that the proposed methodology presents acceptable solutions for the buses power mismatches and the results are also better or as good as those obtained through Newton-Raphson method. The proposed methodology computational implementation is simpler than the Newton-Raphson method computational implementation. It is proposed for the future works an improvement in the proposed algorithm in order to achieve lower mismatches in the cases which they were about 10^{-3} or 10^{-4} and also to evaluate its applicability in cases where the traditional methods fail or have difficult convergence.

Acknowledgments. The authors would like to thank CNPq, CAPES, and FAPEMIG - Brazilian research funding agencies, for the research scholarships, which supported this work.

References

1. Paucar, V.L., Rider, M.J.: On The Use of Artificial Neural Networks for Enhanced Convergence of the Load Flow Problem in Power Systems. In: Proceedings of Intelligent Systems Applications to Power Systems, ISAP 2001, vol. 1, pp. 153–158 (2001)
2. Stott, B.: Review of Load Flow Calculation Methods. IEEE Proceedings 62, 916–929 (1974)
3. Elgerd, O.I.: Electric Energy Systems Theory: An Introduction. McGraw-Hill Publishing Co. Ltd, New York (1975)
4. Esmin, A.A.A., Lambert-Torres, G., Zambroni de Souza, A.C.: A Hybrid Particle Swarm Optimization Applied to Loss Power minimization. IEEE Transactions on Power Systems 20(2), 859–866 (2005)
5. Swarup, K.S.: Swarm Intelligence Approach to the Solution of Optimal Power Flow. J. Indian Inst. Sci. 86, 439–455 (2006)
6. Acharjee, P., Goswami, S.K.: Chaotic Particle Swarm Optimization Based Reliable Algorithm to Overcome the Limitations of Conventional Power Flow Methods. In: Proceedings of Power Systems Conference and Exposition, Digital Object Identifier: 10.1109/PSCE.2009.4839945 (2009)
7. Wang, C., Liu, Y., Pan, X.X.: Load Flow Calculation of Integrated Shipboard Power System Based on Particle Swarm Optimization Algorithm. In: Proceedings of the International Conference on Test and Measurement, ICTM 2009 (2009)

8. Oumarou, I., Jiang, D., Yijia, C.: Particle Swarm Optimization Applied to Optimal Power Flow Solution. In: Proceedings of 2009 Fifth Conference on Natural Computation, vol. 3, pp. 284–288 (2009)
9. Lambert-Torres, G., Martins, H.G., Coutinho, M.P., Salomon, C.P., Matsunaga, F.M., Carminati, R.A.: Comparison between PSO and GA in System Restoration Solution. In: Proceedings of 15th Conference on Intelligent System Applications to Power Systems, ISAP 2009, pp. 1–6. IEEE Press, Curitiba (2009)
10. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
11. Lambert-Torres, G., Martins, H.G., Coutinho, M.P., Salomon, C.P., Filgueiras, L.S.: Particle Swarm Optimization Applied to Restoration of Electrical Energy Distribution Systems. In: Kang, L., Cai, Z., Yan, X., Liu, Y. (eds.) ISICA 2008. LNCS, vol. 5370, pp. 228–238. Springer, Heidelberg (2008)
12. Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., Nakanishi, Y.: A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems Considering Voltage Security Assesment. IEEE Transactions on Power Systems 15(4), 1232–1239 (2001)
13. Wood, A.J., Wollenberg, B.F.: Power Generation, Operation and Control, 2nd edn. John Wiley & Sons Ltd, Chichester (1996)
14. Onate, P.E., Ramirez, J.M.: Optimal Power Flow Solution with Security Constraints by a Modified PSO, Power Engineering Society General Meeting. In: 2007 IEEE, Digital Object Identifier: 10.1109/PES.2007.386005 (2007)
15. Kennedy, J., Eberhart, R.: The Particle Swarm: Social Adaptation of Knowledge. In: Proc. IEEE International Evolutionary Computation ICEC 1997, Indianapolis, USA, pp. 303–308 (1997)
16. Beasley, D., Bull, D., Martin, R.: An Overview of Genetic Algorithms: Part 1, Fundamentals, Technical Report, Inter-University Committee on Computing (1993)
17. Goldberg, D.E., Holland, J.H.: Genetic Algorithms and Machine Learning: Introduction to the Special Issue on Genetic Algorithms. Machine Learning 3 (1988)
18. Davies, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
19. Lambert-Torres, G., Martins, H.G., Coutinho, M.P., Borges da Silva, L.E., Matsunaga, F.M., Carminati, R.A., Cabral Neto, J.: Genetic Algorithm to System Restoration. In: 2009 World Congress on Electronics and Electrical Engineering, WCEEENG 2009, Cairo, Egypt (2009)

Simulation of Routing in Nano-Manipulation for Creating Pattern with Atomic Force Microscopy Using Hybrid GA and PSO-AS Algorithms

Ahmad Naebi[1], Moharam Habibnejad Korayem[2], Farhoud Hoseinpour[3], Sureswaran Ramadass[3], and Mojtaba Hoseinzadeh[1]

[1] Qazvin Islamic Azad University, Qazvin, Iran
[2] College of Mechanical Engineering, Iran University of Science & Technology,Tehran, Iran
[3] National Advanced IPv6 Centre, Universiti Sains Malaysia, Penang, Malaysia
Ahmad.Naebi@gmail.com, Hkorayem@iust.ac.ir, Farhoud@nav6.usm.my, sures@nav6.usm.my, Mojtaba.h1361@yahoo.com

Abstract. Avoiding collision of nano-particles during manipulation operations and selecting the best route and lowest Atomic Force Microscopy (AFM) movement are major concerns in the area of nano-space. To apply the lowest force on the cantilever from fluid environment forces, we try to minimize AFM movements. Our proposed method calculates the optimum routing for AFM probe movement for nano-particles transmission using hybrid GA (Genetic algorithm) and PSO-AS (Particle Swarm Optimization- Ant System) simulates it in various type of medium. We consider the collision of the probe with minor barriers. An optimized AFM path minimizes the time and energy required for nano-particle manipulation. For movement of the nano-particles, we seek an efficient probe pattern. A second goal is to transfer the nano-particles without undesired collision. The optimum routing method will increase the speed of the process. Our proposed model, utilizes both Mathematical and Matlab software to simulate the process.

Keywords: Simulation, Routing, Nano-manipulation, GA and PSO-AS, Creating Pattern, Atomic Force Microscopy.

1 Introduction

Nanoparticles are suitable units to be used in nano-structures. Also, we can use AFM manipulator in order to create patterns of nano-particles. Furthermore, AFM capability in imaging of the surfaces and in nano-particle manipulation makes it an effective tool to be exploited in creating nano-particles patterns [1]. AFM is used as a simple nano-manipulator for guiding nano-particles movement on surfaces. In this study, we have defined nano-particles as having a radius R_P and with the capability of being absorbed in substrate and ridden by an AFM probe under particular liquid environmental conditions. In order to ride the particles, the tip of the probe must be in contact with the particle. So, for the safety of this contact under particular conditions, a basic jump, Z_{P0}, should be verified through system feedback. Different steps of nano-manipulation of a particle movement are shown in Figure.1 [2, 3]. Figure 1 shows states of manipulation using probe of AFM in contact mode.

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 606–615, 2011.
© Springer-Verlag Berlin Heidelberg 2011

Using an optimal routing algorithm, we can manipulate nano-particles quickly and, without any collision, can create patterns in various types of environments. In this article, we'll focus on creating patterns in almost any environment. At first nanoparticles are randomly distributed in the environments. Creating patterns for routing is performed by using hybrid GA and PSO-AS without any collision, hence no nano-particle collides with another.

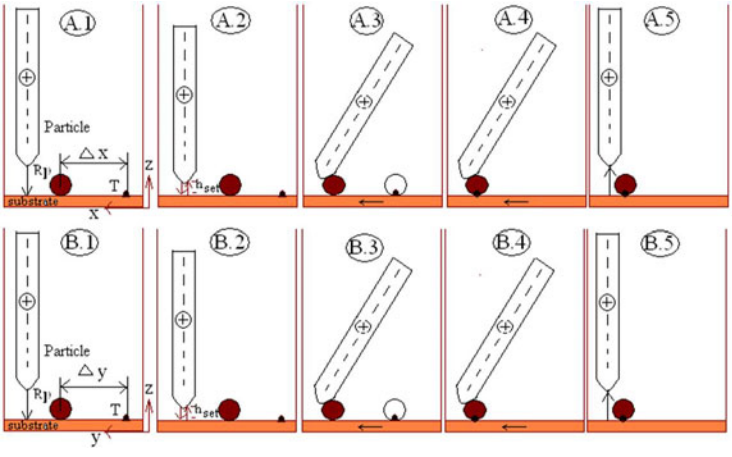


Fig. 1. The strategy of pushing with AFM in Contact model[3]

2 View of Dynamic Modeling

In nano-manipulation processes, the probe is considered to be a cylinder [4]. The involved forces between probe and cantilever and also between probe and nano-particle are illustrated in Figures 2 respectively. As displayed in these figures, there are 3 forces applied on the cantilever and the probe: Fx, Fy, and Fz.

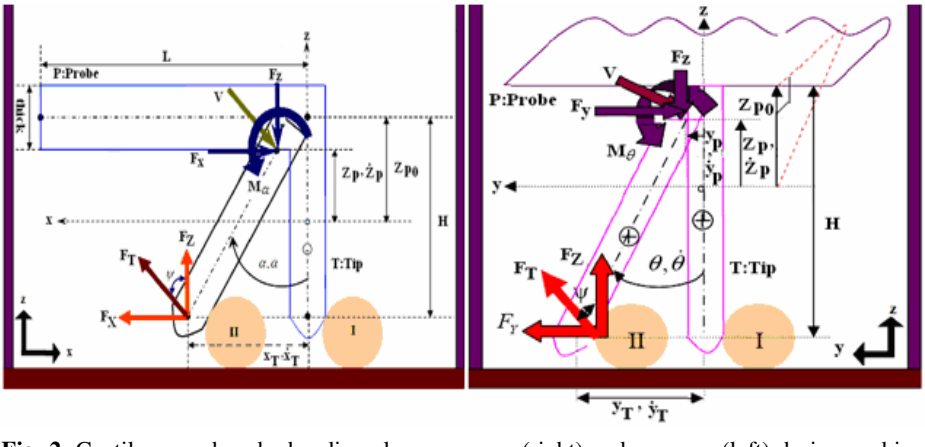


Fig. 2. Cantilever and probe bending along y-z axes (right) and x-z axes (left) during pushing nano objects in air [4]

Dynamic equations are developed based on the free body diagram (FOB) of pushing system, including AFM cantilever and probe, nano-particle and substrate. Fig. 3 presents the FOB of probe of AFM in contact with particle. Total steps of manipulation under fluid conditions are illustrated in figure 3 All of the forces including fluid, frictional, and adhesive forces on the tip of the cantilever are calculated [5, 6].

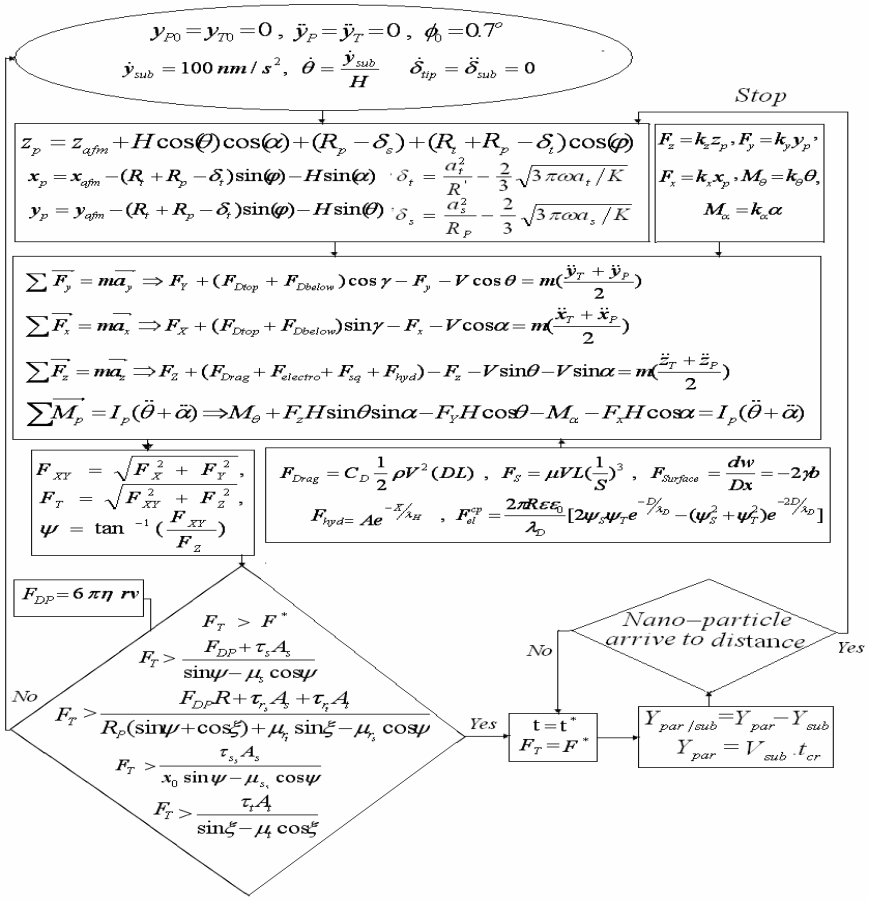


Fig. 3. Flowchart of dynamic modelling algorithm in pushing nano-particle [3]

3 GA and PSO Algorithms

PSO is an optimization algorithm innovated by Kennedy and Eberhart in 1995 and inspired by swarm intelligence. A swarm is a group of particles with each particle containing position and velocity vectors [7, 8, 9]. The “basical” PSO equation, where the position and velocity represent physical attributes of the particles, is represented

by (1) and (2) [9]. Calculating a single Particle's New Velocity (1) and Moving of a single Particle in a swarm (2)

$$V_{t+1} = C_1 V_t + C_2(P_{ig} - X_t) + C_3(P_{\forall g} - X_t) \quad (1)$$

$$X_{t+1} = X_t + V_{t+1} \quad (2)$$

The Genetic Algorithm (GA) has been suggested using Darwin's evolution theory and a searching process based on the natural selection and genetics lows. GA is considered as one of the best algorithms for optimization problems. [10, 11]. Often, all of simple GA consists of three operations: Selection, Genetic Operation, and Replacement [10]. The operations which are used in genetic algorithm are crossover and mutation.

3.1 Description of Hybrid Algorithms

Expressing a route: Any-member of the population could be considered as a route. A route consists of a set of nano-particles which forms our patch. These nano-particles are to migrate from their current location to another destination. A group of nano-particles in a patch form a route between the source and destination, instead of transferring only one nano-particle through a route. In fact instead of covering the patch by nano-particles, probe of AFM routes all through the patch. Some of the nano-particles and nodes are the destinations of other nano-particles. Figure 4.A shows an example of members used in this method, which included information of nano-particle's position and their potential destination.

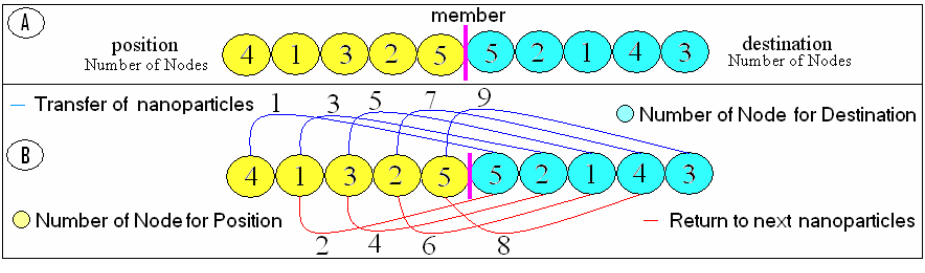


Fig. 4. An example of member (including nano-particles with yellow color and place for transfer nano-particles with green color)

Coding members: Assume an environment with 5 nodes to detect the environment. First five nodes are used for nano-particles position and five next nodes are used as their destinations. For example nano-particles has been transferred as following (4 to 5,1 to 2, 3 to 1, 2 to 5,5 to 3). As it is shown in figure 4.B, red lines are nano-particle positions and blue lines are their destinations.

Structure of PSO-AS [9]:

When a connection request arrives at the ingress node, a predetermined number of particles are created. Each particle in the algorithm contains a random general route that presents the location of the nano-particle using a specific route from source to destination. So, after initial random assignment of the routes, each particle will have a position. Equation 2 is used to update the fitness value of all the particles. The particle

with the lowest fitness value in the current iteration is marked as global best. The lowest value met by the particle through its travel from start to current iteration is marked as a local best [9].

Every particle will then have its new velocity calculated. This will take it to the new position (i.e. establish a new route with new source and its potential destination) given by equation 3. The velocity is represented as a sequence of nodes in a particular order, and is calculated according to equations 3 and 4, in a step-by-step manner as in Ant Systems. Equation 3 will find just one member (next node to go) of the new velocity sequence, in a single step [9].

$$V_{t+1} = N_{k+1} + N_{k+2} \tag{3}$$

$\forall k = 1$ to D-2 where N=Source Node and N=Destination Node.

$$N_a = \begin{cases} P_{\forall\ g,\ t} - \omega_t & \text{if} \quad \eta \le C_3 \\ P_{i\ g,\ t} - \omega_t & \text{if} \quad \eta > C_3 \ \ \& \ \ \eta \le C_2 + C_3 \\ Rand \quad Selection & \text{if} \quad \eta > C_2 + C_3 \ \ \& \ \ \eta \le C_1 + C_2 + C_3 \\ & where \quad C_1 + C_2 + C_3 = 100\ \% \end{cases} \tag{4}$$

$$where \ \ldots \quad a = k + 1, k + 2$$

After calculating the new velocity Vt+1, each particle will update its current position Xt by using (2). When the particle has moved to a new position, the fitness value of the particle is updated according to (5) [9].

$$F(x) = \beta*(\text{hop-count}) + (1-\beta)(F_w/W) \tag{5}$$

β is an algorithm parameter. F_w is the number of free nano-particles and distances available on the route. W is the calculated total number of nano-particle distances. Table I summarizes all of parameters used in PSO-AS [9].

Describing the algorithm[12]:

We select first node of the member as first node of the route. Then we create a random number for selecting destination node. If the random number be less than C3, we'll select destination node of the global best. A destination node will be selected from global best (best member) which is existed in the global best (have not selected before, and are peer to peer). If the random number is more than C3 and less than C2+C3, we will select destination node of the local best. A destination node will be selected from the local best which is existed in the local best (have not selected before, and are peer to peer). If destination have not selected in above conditions, we will randomly select a destination node which have not selected before. For the next node above steps will be repeated but source and destination positions will be changed. This method will be used for all of nodes including nano-particle positions and destinations.

Structure of Genetic:

Crossover: First, we select a random number. If it be less than a const, Crossover will done in left portion of selected position in a chromosome. If it be more than const, Crossover will done in right portion of selected position in a chromosome. Crossover in left portion of a member don't use nodes of same member that are in right portion of it, only the nodes are exchanged that are in left portion. Crossover in right portion of a member don't use nodes of same member that are in left portion it, only the nodes are exchanged that are in right portion (Fig. 5.A) . Crossover was done for position and destination of members. Position and destination of member is showed with yellow and blue color in figure 5.A.

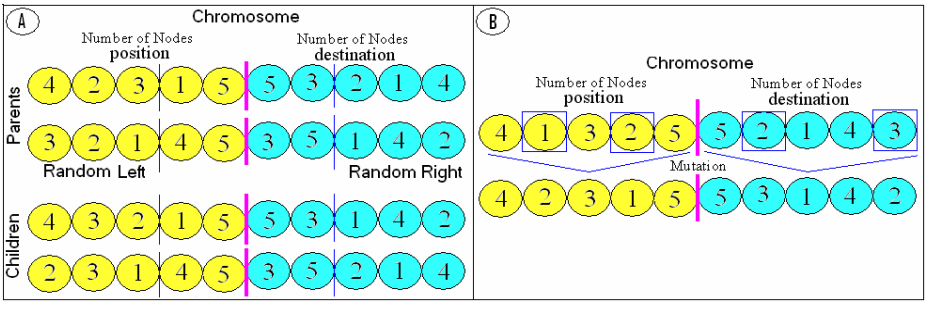


Fig. 5. A) Crossover of self-left nano-particles (nodes of yellow color). B) Mutation do in a member(in position and destination).

Mutation: First, we select a random number. If, it be less than 5%, mutation will be done. In mutation, we select two number as random and exchange them. We do the mutation separately for position and destination nano-particles. (Fig.5.B)

Four parts in both algorithms are identical:

1. Route length: In this step route length is calculated for each member. Then all of them will be saved in 1st column of the fitness array.

2. Total collisions: If a nano-particle is too close to another nano-particle moving route and the distance is less than a certain value, collision will be occurred. For each member total collisions are calculated. Then all of them will be saved in 2nd column of the fitness array.

3. Selecting the global best of current state: If two or more of members have equal number of collisions and number of collisions is the lowest among other members, we will select the route with minimum route length. But, if we have just one member with fewer collisions, we will select it, because having fewer collisions has high priority of less route length (Fig. 6.Left).

4. Selecting global best: We have the global best which is the best member till former stage. If the best member of this stage is better than the global best, we will replace the global best with it. There are three cases for this replacement you can see in fig. 7.

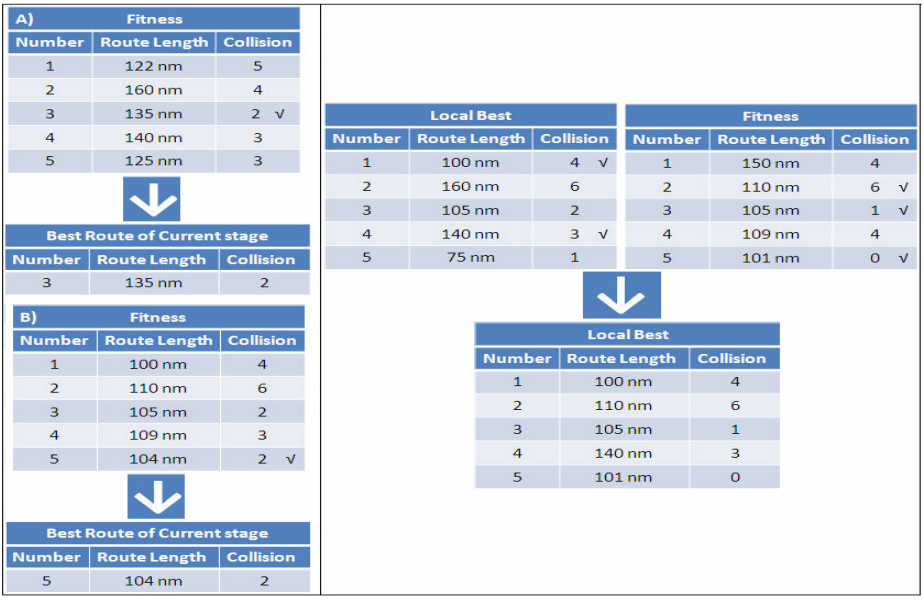


Fig. 6. Select best member of fitness in a stage and Change local best if possible

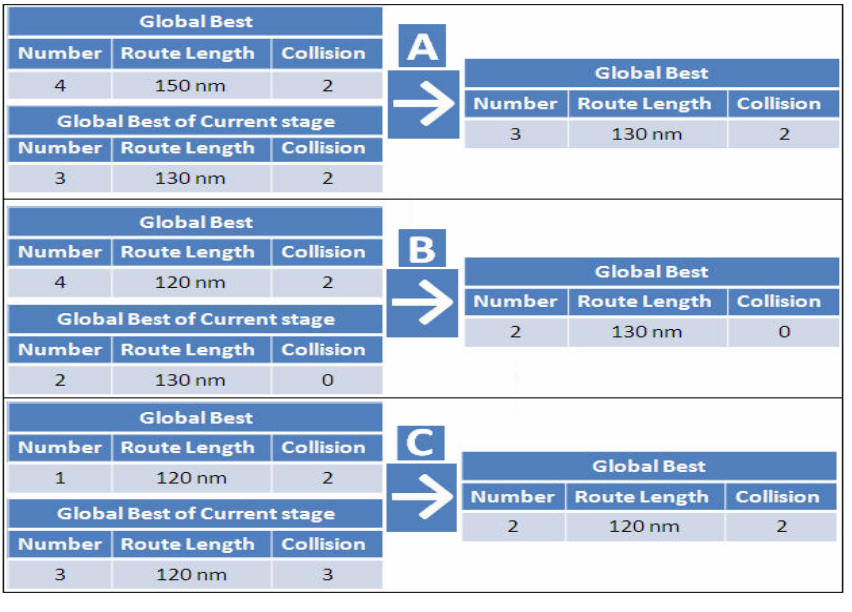


Fig. 7. Select best global best

Selecting the local best: If the local best of each member is worse than the created route for each member in this stage, we will replace the local best with created route as shown in fig. 6.right.

Collision detection: Shorter path between two nanoparticles is obtained from the following equation:

$$Dic\ _\ Crash\ = \sqrt{(Y_{CrashPo\ \text{int}} - Y_{otherparti\ cle})^2 + (X_{CrashPo\ \text{int}} - X_{otherparti\ cle})^2} \quad (6)$$

4 Simulation in Mathematical

We have designed a simulation model for transferring nano-particles in nano-scale in Mathematica software. This simulation covers all parts of Figure 1 and all formulas of section 2 used in the simulation.Figures 8 and 9 are a view of this simulation. At first we locate the probe of AFM in (20μm,20μm,20μm). Then it is moved to first nanoparticle. After creating the pattern the prove go back back to First place.

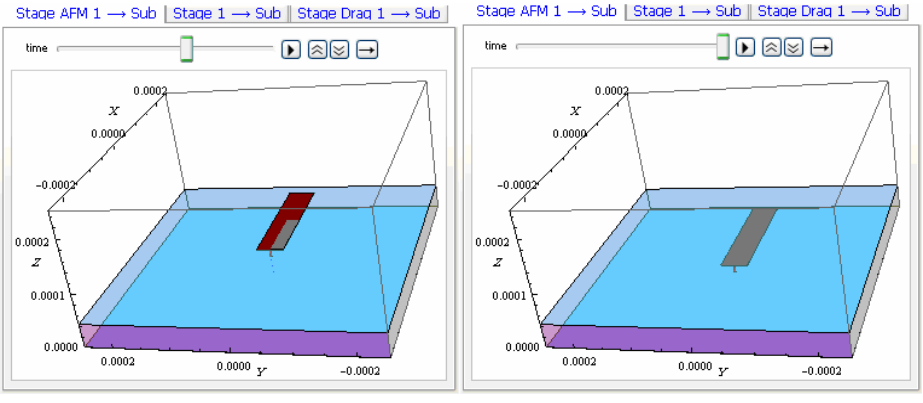


Fig. 8. Veiw of simulation in micro-scale in water

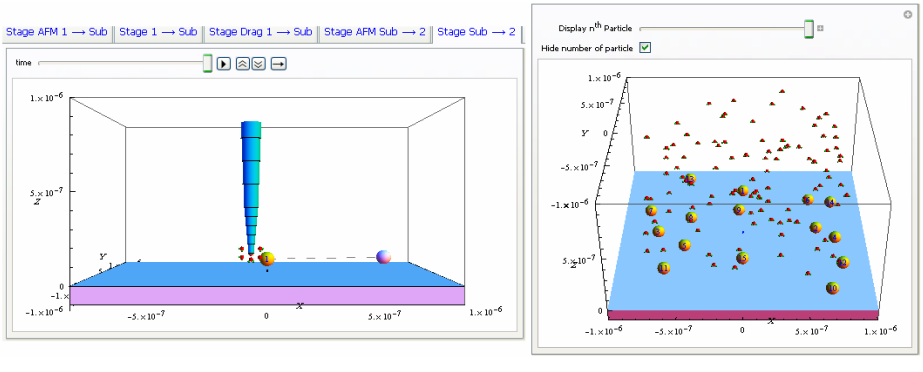


Fig. 9. Veiw of simulation in nano-scale in air and water

5 Simulation of Creating Pattern

First, we distributed 15 nano-particles in small range of a 1000×1000 nm environment. Nano-particles were created in Mathematica and the routing algorithm was simulated in a Matlab environment. We manipulated nano-particles to create a pattern. You can see a pattern with shape of (A) as well as the state of routing in Figure 10. Secod, we distributed 10 nano-particles in the same environment. You can see a pattern with shape of (B) as well as the state of routing in Figure 11.

For finding best routing, we have run 100 iteratoin with 200 member for 10 nano-particles and have run 500 iteratoin with 200 member for 15 nano-particles. But

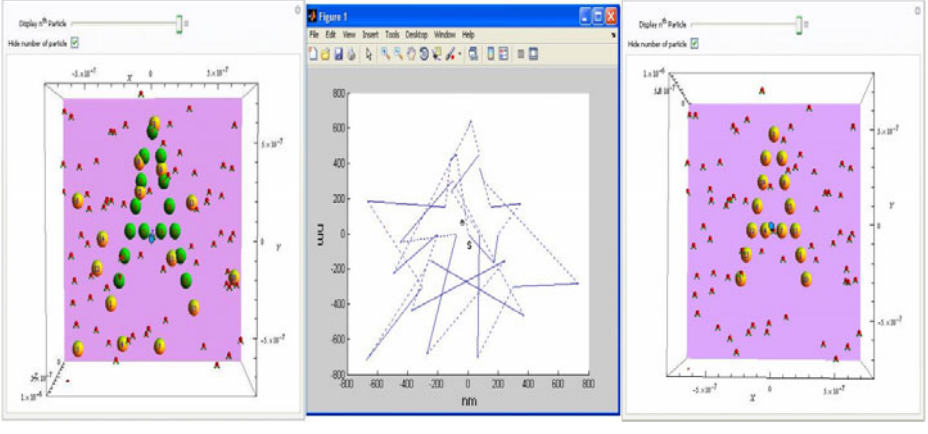


Fig. 10. manipulation nanoparticles for creating pattern(A)

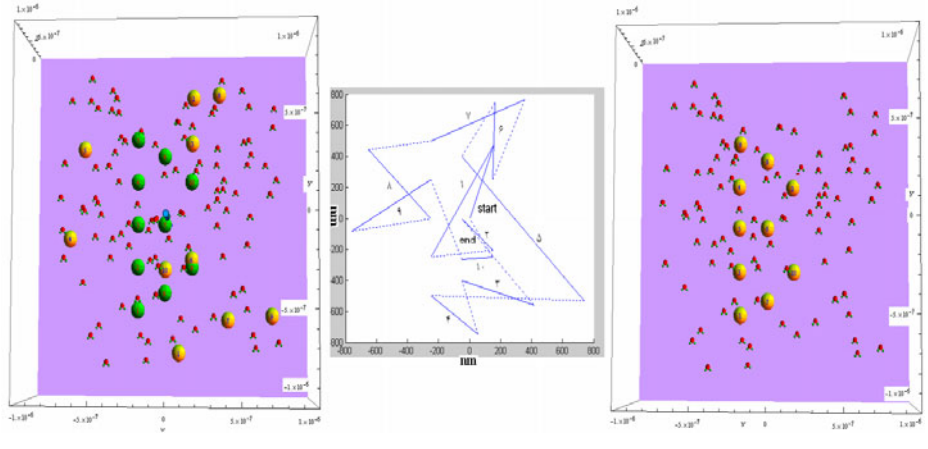


Fig. 11. manipulation nanoparticles for creating pattern(B)

when nano-particles of a pattern are more than 15 nano particle, total iteration will be more. In fact total iteration with different nano particle is different. In this study, we have run once GA and once PSO-AS until finished total iterations.

6 Conclusions

Using our model, transferring of nano-particles for creating the patterns is performed faster using optimized routes for the nano-particles. The advantage of our method is collision avoidance, so AFM doesn't need to image again. In this manner, we further reduce the time required for creating patterns. The simulation also helps us in understanding the pattern creation process and, therefore, the micro and nano-scale. This method of creating patterns is performed in less time and with less applied force. Nano-particles will find best location for transfer. Future work will be done this way: Also in order to create the new nano-instrument in air or fluid environments, our simulation models can be used to facilitate the experiment and their functionality without damaging the real instruments.

References

1. Fotiadis, D., Scheuring, S., Muller, S.A.: Imaging and Manipulation of Biological Structures with the AFM. Micron 33, 385–397 (2002)
2. Sitti, M., Hashimoto, H.: Force Controlled pushing of nano particles: modeling and experiments. IEEE/ASME Trans. on Mechatronics 5, 199–211 (2000)
3. Korayem, M.H., Naebi, A., Esmaeilzadeha, S., Shahri, A.M.: Tele-operated Nanomanipulation in Liquid Environment with Atomic Force Microscopy. In: Proceedings of the 4th Asia International Symposium on Mechatronics (AISM 2010), AISM 2010 Organizers (2010) ISBN: 978-981-08-7723-1
4. Sitti, M., Tafazzoli, A.: Dynamic Behavior And Simulation Of Nanoparticle Sliding During Nanoprobe-Based Positioning. In: Proceedings Of IMECE 2004 ASME International Mechanical Engineering Congress Anaheim, CA, November 13-19 (2004)
5. Dr Erts. Study of the nano scale contacts with the help of combine TEM-AFM technique and theoretical MD-TM calculation. Department of Engineering, Physics, and Mathematics, Mid Sweden University
6. Sitti, M., Hashimoto, H.: Force controlled pu shing of nanoparticles: modeling and experiments. IEEE/ASME Trans on Mechatronics (2000)
7. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
8. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
9. Hassan, A., Phillips, C., Pitts, J.: Dynamic Routing and Wavelength Assignment using Hybrid Particle Swarm Optimization for WDM Networks, © PGNet (2007) ISBN: 1-9025-6016-7
10. Tang, K.S., Man, K.F., Kwong, S., He, Q.: Genetic Algorithms and their Applicaition. IEEE Signal Processing Magazine (1996)
11. Nagib, G., Ali, W.G.: Network Routing Protocol using Genetic Algorithms. International Journal of Electrical & Computer Sciences IJECS-IJENS 10(02)

Neural Fuzzy Forecasting of the China Yuan to US Dollar Exchange Rate — A Swarm Intelligence Approach

Chunshien Li[1], Chuan Wei Lin[1], and Hongming Huang[2]

Laboratory of Intelligent Systems and Applications
[1] Department of Information Management
[2] Department of Financial Economics
National Central University, Taiwan, R.O.C.
jamesli@mgt.ncu.edu.tw

Abstract. Exchange rate fluctuation has a significant effect on the risk of marketing business, economic development and financial stability. Accurate prediction for exchange rate may reduce commercial and economic risk arisen by exchange rate fluctuation. In this study, we propose an intelligent approach to the forecasting problem of the CNY-USD exchange rate, where a neuro-fuzzy self-organizing system is used as the intelligent predictor. For learning purpose, a novel hybrid learning method is devised for the intelligent predictor, where the well-known particle swarm optimization (PSO) algorithm and the recursive least squares estimator (RLSE) algorithm are involved. The proposed learning method is called the PSO-RLSE-PSO method. Experiments for time series forecasting of the CNY-USD exchange rate are conducted. For performance, the intelligent predictor is trained by several different methods. The experimental results show that the proposed approach has excellent forecasting performance.

Keywords: time series forecasting, neuro-fuzzy system (NFS), particle swarm optimization (PSO), recursive least-squares estimator (RLSE), hybrid learning, self-organization.

1 Introduction

Exchange rate systems can be divided into the categories of floating rate and fixed rate. Before the collapse of Bretton Wood System [1], banks adopted the fixed exchange rate system. After 1973, many countries have turned to adopt floating exchange rate. Since the floating exchange rate is changeably fluctuated, it may induce impact and risk on international trade, economic investment, and financial stability. Changes of exchange rate may come up with opportunity and risk. How to reduce the risk is a very interesting research topic. Many factors can affect exchange rates, for example, inflation rates, government intervention and economic growth [2]. The historical records in time sequential order of an exchange rate can be viewed as a time series [3]. Based on a prediction model, time series forecasting is to forecast future trend or change, based on past known observations.

Several approaches have been presented to predict exchange rate in literature [4]-[7]. Neural fuzzy system (NFS) approach [8] is one of the major methods. Many machine-

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 616–625, 2011.
© Springer-Verlag Berlin Heidelberg 2011

learning algorithms can be used to adjust the parameters of a NFS for application, such as, particle swarm optimization (PSO) [9] [10] [11], genetic algorithm (GA) [12], simulated annealing [13], simplex method [14], and downhill climbing method [15]. In this study, we use the theory of neuro-fuzzy system (NFS) to design an intelligent predictor for the time series forecasting problem of the CNY-USD exchange rate. In the NFS-based intelligent predictor, there are several fuzzy If-Then rules. The fuzzy rule can be divided into the Mamdani type and the Takagi-Sugeno (T-S) type [16] [17]. The difference between them lies on the Then-parts of the fuzzy rules. The Then-part of a T-S fuzzy rule is a function of inputs while a Mamdani fuzzy rule has a fuzzy set for its Then-part. Because T-S fuzzy rules possess greater capability of nonlinear functional mapping for applications than Mamdani fuzzy rules, we use T-S fuzzy rules for the NFS-based predicting approach in this study. For the formation of the NFS-based predictor in the study, there are two learning stages, the structure learning stage and the parameter learning stage. For the structure learning stage, we use a Fuzzy C-Means (FCM) based clustering method [18] to automatically determine the optimal number of fuzzy rules for the NFS-based predictor. Note that clusters generated the clustering method are regarded as fuzzy rules, based on the concept of input space partition. For the parameter learning stage, we develop a hybrid learning method, called the PSO-RLSE-PSO method, to fine-tune the parameters of the predictor. The hybrid learning method includes the well-known particle swarm optimization (PSO) algorithm and the recursive least squares estimator (RLSE) algorithm [16] [19]. In a hybrid way, the PSO algorithm updates the premise parameters of If-Then fuzzy rules, and the RLSE algorithm adjusts the consequent parameters. Afterward, the premise parameters are fixed, and the PSO algorithm is used again to update the consequent parameters, based on the result by the RLSE. With the PSO-RLSE-PSO hybrid learning algorithm, the intelligent can make better prediction accuracy.

In Section 2, the proposed methodology is specified, including the theory of NFS, the hybrid learning method and the FCM-based clustering algorithm for self-organization of the NFS-based predictor. In Section 3, experiments for the China Yuan (CNY) to US Dollar (USD) exchange rate forecasting by the proposed approach and the compared approaches are conducted, and the experimental results are given. Finally, a discussion is given is given and the paper is concluded.

2 Methodology of the Proposed Approach

2.1 Theory of Neuro-Fuzzy System (NFS)

The theory of fuzzy sets and fuzzy logic can be used to transform the experience of experts and knowledge into fuzzy If-Then rules, which are usually with uncertain information and imprecise expression. Fuzzy systems using this excellent property are usually known as an excellent problem-solving paradigm. Artificial neural networks have strong learning capability and they can adaptively learn to find hidden information and they have made great progress for extreme learning machines [20] [21]. Both of fuzzy systems and neural networks are universal approximator, by which any function can be approximated to any accuracy theoretically [22] [23].Integrating the advantages of fuzzy system and neural network, a NFS [10] [16]

is a fuzzy inferential neural system, which can handle with complex issues. Suppose that there are *K* Takagi-Sugeno (T-S) type fuzzy rules [17] in a NFS. The *i*th fuzzy If-Then rule can be expressed as follows.

$$\begin{aligned} &\text{Rule } i\text{:} \quad \text{IF } x_1 \text{ is } s_1^i\big(h_1(t)\big) \text{ and } \ldots \text{and } x_M \text{ is } s_M^i\big(h_M(t)\big) \\ &\qquad\qquad \text{Then } z^i = a_0^i + a_1^i h_1(t) + \cdots + a_M^i h_M(t) \end{aligned} \tag{1}$$

for i=1,2,…,K, where $\mathbf{H}(t)=[h_1(t)\ h_2(t)\ \ldots\ h_M(t)]^{\mathrm{T}}$ is the crisp input vector to the NFS at time t, $\{x_i,\ i=1,2,\ldots,M\}$ are the input linguistic variables, $\{s_j^i,\ j=1,2,\ldots,M\}$ are the premise fuzzy sets of the ith fuzzy rule, z^i is the output, and $\{a_j^i,\ j=0,1,\ldots,M\}$ are the consequent parameters. Note that in the study the inputs are from historical data of exchange rate time series. The fuzzy inference of fuzzy system can be cast into neural-net structure with five layers to become the NFS [10] [16] [24] [25]. The explanation for the structure of NFS layer by layer is specified as follows. **Layer 1:** This layer is called the fuzzy-set layer. There are several nodes in the layer. Each node of the layer represents a fuzzy set in an input universe, by which the measured crisp input is transformed into fuzzy value. Each node output is a membership degree. In the study, fuzzy sets are designed using the Gaussian type membership function, given below.

$$\text{Gaussian}(h) = \exp\left(-\frac{1}{2}\left(\frac{h-m}{\sigma}\right)^2\right) \tag{2}$$

where h is a base variable, $\{m, \sigma\}$ are the mean and spread. **Layer 2:** This layer is called the firing-strength layer. The premise parts of fuzzy rules are formed in this layer. The nodes of the layer perform product operation (for t-norm operation) to calculate the firing strengths of the fuzzy rules, which are written as $\omega^i(t) = \prod_{j=1}^{M} s_j^i\left(h_j(t)\right)$, i=1,2,…,K. Note that $\{s_j^i,\ j=1,2,\ldots,M\}$ are the premise fuzzy sets of the i-th fuzzy rule specified in (1) and designed in (2) in the study. **Layer 3:** This layer is normalization layer. The nodes in the layer perform the process of normalization for the firing strengths of the fuzzy rules, given below.

$$\lambda^i(t) = \frac{\omega^i(t)}{\sum_{i=1}^{k} \omega^i(t)} \tag{3}$$

Layer 4: This layer is called the consequent layer. The nodes perform normalized consequents of all the fuzzy rules. Each node output represents a normalized consequent of fuzzy rule, given below.

$$\lambda^i(t)z^i(t) = \lambda^i(\mathrm{t})\left(a_0^{\mathrm{i}} + \sum_{j=1}^{M} a_j^i h_j(t)\right) \tag{4}$$

for i=1,2,…,K. **Layer 5:** This layer is called the output layer. The number of nodes is equal to that of the NFS outputs. In the study, we need one node only, which combines the node outputs of Layer 4 to produce the forecast for the CNY-USD exchange rate by the NFS output $\zeta(t)$ below.

$$\zeta(t) = \sum_{i=1}^{k} \lambda^i(t) z^i(t) = \sum_{i=1}^{k} \frac{\prod_{j=1}^{M} s_j^i\left(h_j(t)\right)}{\sum_{i=1}^{k}\prod_{j=1}^{M} s_j^i\left(h_j(t)\right)} \left(a_0^i + \sum_{j=1}^{M} a_j^i h_j(t) \right) \tag{5}$$

2.2 Clustering for Self-organization of the NFS Predictor

A Fuzzy C-means (FCM) based clustering method is used the self-organization of the intelligent NFS based predictor, by which the optimal number of fuzzy rules are automatically determined. The FCM algorithm, derived from K-means algorithm, was first proposed by Dunn and Bezdek [26]. Although FCM is better than K-means, but it still needs to enter a preset number of clusters for clustering. Haojun et al. improved this drawback and proposed the FCM-based splitting algorithm (FBSA) [18], which is based on the FCM method and a cluster validity index [27]. The FBSA can select the optimal number of clusters in between two preset positive integers, C_{min} and C_{max} which are the minimal and maximal numbers of clusters for the algorithm. The FBSA algorithm is given below. Step1: Preset C_{min} and C_{max}. Step 2: initialize C_{min} cluster centers ($\mathbf{V}$). Step 3: Do for-loop $c = C_{min}$ to C_{max}; 3.1: apply the FCM algorithm to update the membership matrix $\mathbf{U}$ and the cluster centers, $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_c\}$, where $\mathbf{v}_j = [v_{j,1}, v_{j,2}, \cdots, v_{j,Q}]^{\mathrm{T}}$ is the j-th cluster center; 3.2: test for convergence; if no, go to 3.1; 3.3: compute a validity value $V_d(c)$; 3.4: compute a score $S(i)$ for each cluster; split the worst cluster. Step 4: Compute the optimal number c_f such that the cluster validity index $V_d(c_f)$ is optimal. The validity index $V_d(.)$ is defined as follows.

$$V_d(\mathbf{U}, \mathbf{V}, c) = Scat(c) + \frac{Sep(c)}{Sep(C_{max})} \tag{6a}$$

$$Scat(c) = \frac{\frac{1}{c}\sum_{i=1}^{c}\|\sigma(\mathbf{v}_i)\|}{\|\boldsymbol{\sigma}(\mathbf{X})\|} \tag{6b}$$

$$Sep(c) = \frac{D_{max}^2}{D_{min}^2} \sum_{i=1}^{c} \left(\sum_{j=1}^{c} \|v_i - v_j\|^2 \right)^{-1} \tag{6c}$$

where $\boldsymbol{\sigma}(\mathbf{X}) = [\sigma(\mathbf{X})_1, \sigma(\mathbf{X})_2, \cdots, \sigma(\mathbf{X})_Q]^{\mathrm{T}}$, $\sigma(\mathbf{X})_p = \frac{1}{n}\sum_{k=1}^{n}(x_{k,p} - \bar{x}_{k,p})^2$, $D_{min} = min_{i \neq j}\|\mathbf{v}_i - \mathbf{v}_j\|$, $D_{max} = max_{i \neq j}\|\mathbf{v}_i - \mathbf{v}_j\|$ and n is the total number of data vectors in a given data set. Note that $Scat(c)$ represents the compactness among obtained clusters, whose range is between 0 and 1, and $Sep(c)$ represents the separation among the clusters.

2.3 Parameter Learning with PSO-RLSE-PSO Hybrid Learning Method

Particle Swarm Optimization (PSO) [9] is a swarm-based optimization algorithm, motivated by the food searching behavior by bird flocking or fish schooling in a society-oriented pattern. For a bird swarm, each bird is regarded as a particle whose location is viewed as a candidate solution to the problem. The search movement of the ith particle of the swarm is affected by two factors, the particle's best location in its

experience and the swarm's best location during the search, denoted by $\mathbf{Pbest}_i$ and **Gbest**, respectively. Assume that the search space is with Q dimensions. The location and velocity of the ith particle at time t are denoted as $\mathbf{X}_i(t)$ and $\boldsymbol{\psi}_i(t)$, respectively. The PSO algorithm is given by the following equations.

$$\boldsymbol{\psi}_i(t+1) = w \times \boldsymbol{\psi}_i(t) + c_1 \times \xi_1 \times \left(\mathbf{Pbest}_i - \mathbf{X}_i(t)\right) + c_2 \times \xi_2 \times (\mathbf{Gbest} - \mathbf{X}_i(t)) \tag{7a}$$

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \boldsymbol{\psi}_i(t) \tag{7b}$$

where w in the inertia factor, $\{c_1,c_2\}$ are the learning rates, $\{\xi_1,\xi_2\}$ are random uniformly between 0 and 1. The procedure of PSO algorithm is given as follows. Step 1: initialize the position and velocity of particles. Step 2: calculate fitness for current position with fitness function $f(.)$. Step 3: update position and velocity of each particle. Step 4: If $f(\mathbf{Pbest}_i)$ gets improved, then update $\mathbf{Pbest}_i$. Step 5: If $f(\mathbf{Pbest}_i) < f(\mathbf{Gbest})$, then update **Gbest**. Step 6: If termination condition is met, stop. Otherwise, go to Step 3 and the procedure continues.

In the general least squares estimation (LSE) problem, a linear model can be described as follows [19].

$$y = \theta_1 f_1(\mathbf{u}) + \theta_2 f_2(\mathbf{u}) + \cdots + \theta_n f_n(\mathbf{u}) + \varepsilon \tag{8}$$

where $\mathbf{u} = [u_1, u_2, \cdots, u_Q]^{\mathrm{T}}$ is the Q-dimensional input to the model, y is the observed data corresponding to the input $\mathbf{u}$, $\{f_1, f_2, \cdots, f_n\}$ are known functions of $\mathbf{u}$, $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_n]^{\mathrm{T}}$ are the parameter vector to be estimated, and ε is the model error. Note that the contents of the vector $\boldsymbol{\theta}$ can be viewed as the consequent parameters of the NFS predictor for the problem of time series forecasting. Observed data can be collected and used as training data for the proposed NFS predictor. The training data set with m data pairs is denoted as $\{(\mathbf{u}_j, y_j), j=1,2,\ldots,m\}$, where $(\mathbf{u}_j, y_j)$ is the j-th data pair in the form of (*input*, *target*). With the training data set, the LSE problem can be written in matrix form, given below.

$$\mathbf{Y} = \mathbf{A}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \tag{9}$$

where $\mathbf{Y} = [y_1, y_2, \cdots, y_m]^{\mathrm{T}}$, $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_m]^{\mathrm{T}}$ and $\mathbf{A}$ is the matrix formed by $\{f_i(\mathbf{u}_j), i = 1,2,\ldots,n \text{ and } j = 1,2,\ldots,m\}$. We denote the kth row of $\mathbf{A}$ and $\mathbf{Y}$ as $[\mathbf{a}_k^{\mathrm{T}}, y_k]$. Recursively, the optimal estimation for $\boldsymbol{\theta}$ can be obtained by the method of recursive least squares estimator (RLSE) [16], given below.

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{a}_{k+1} \mathbf{a}_{k+1}^{\mathrm{T}} \mathbf{P}_k}{1 + \mathbf{a}_{k+1}^{\mathrm{T}} \mathbf{P}_k \mathbf{a}_{k+1}} \tag{10a}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{P}_{k+1} \mathbf{a}_{k+1} (y_{k+1} - \mathbf{a}_{k+1}^{\mathrm{T}} \boldsymbol{\theta}_k) \tag{10b}$$

for k=0,1,…,m-1. Initially, we set $\boldsymbol{\theta}_0$ to zero vector and $\mathbf{P}_0 = \alpha\mathbf{I}$, where α is a large positive number and $\mathbf{I}$ is the identity matrix. The optimal estimator for $\boldsymbol{\theta}$ is obtained after the last pair of training data $(\mathbf{u}_m, y_m)$ is done.

Table 1. Settings for PSO and RLSE

PSO		**RLSE**	
Swarm size	125	Consequent parameters	12
Initial particle position	Random in [0, 1]	$\boldsymbol{\theta}_0$	12×1 zero vector
Initial particle velocity	Random in [0, 1]	$\mathbf{P}_0$	$\alpha\mathbf{I}$, $\alpha = 10^{10}$
$\{w, c_1, c_2\}$	$\{0.8, 2, 2\}$	$\mathbf{I}$	Identity matrix

In the study, a hybrid learning method is devised for the parameter learning of the proposed NFS predictor, based on the result of self-organization by the FBSA clustering method to determine the optimal number of fuzzy rules for the intelligent predictor. There are two stages for the parameter learning. In **stage 1**, the PSO method is used to adapt the premise parameters of the NFS predictor and the RLSE method is use to update the consequent parameters. Both methods cooperate in hybrid way for fast learning purpose in the first stage. In **stage 2**, based on the result of the PSO-RLSE learning in the first stage, the PSO method is used again in the second stage to update the consequent parameters while the premise parameters are kept fixed. This two-stage learning method is called the PSO-RLSE-PSO method, for which the procedure is arranged below.

Step 1. Initialize necessary settings for the PSO and RLSE methods (in **stage 1**).
Step 2. Adapt the premise parameters of the NFS predictor by the PSO algorithm
Step 3. Update the consequent parameters by the RLSE algorithm.
Step 4. Calculate forecast error $e(t) = y(t) - \zeta(t)$, where $y(t)$ is the observed at time t and $\zeta(t)$ is the forecast by the NFS predictor.
Step 5. Calculate root mean squared error (RMSE), based on $\{e(t), t=1,2,\ldots m\}$, as follows.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{m}(e(t))^2}{m}} \tag{11}$$

Step 6. Update **Pbest** (for each particle) and **Gbest** for PSO.
Step 7. If termination condition for the PSO-RLSE learning (**stage 1**) is met, go to **Step 8** to proceed to **stage 2** of the parameter learning. Otherwise, go back to **Step 3**.
Step 8. Adapt the consequent parameters by the PSO (in **stage 2**). Calculate RMSE for all particles of the PSO. Update **Pbest** and **Gbest**.
Step 9. If termination condition for the PSO (**stage 2**) is met, then stop. Otherwise, go back to **Step 8** and the procedure continues.

3 CNY-USD Exchange Rate Forecasting by the Proposed Approach

The dataset recording the daily exchange rate of China Yuan (CNY) to US Dollar (USD) from 2006/01/01 to 2007/12/31 are obtained [28]. The dataset is normalized to [0, 1], from which the first 730 data are used as the training data and the rest 365 data are for testing purpose. For the proposed NFS predictor, the input vector is arranged as $\mathbf{H}(t)=[y(t\text{-}1),\ y(t\text{-}2)]^{\mathrm{T}}$ and the target is $y(t)$, where t is the time index. The FBSA clustering method is used for self-organization of the predictor to automatically determine the optimal number of fuzzy rules. Each cluster generated by the clustering method corresponds to a fuzzy If-Then rule. For the generated clusters, the cluster centers and their spreads in standard deviation are used for the initial settings of the premise fuzzy sets of the fuzzy rues. For parameter learning, the RMSE is used for the cost function. For the FBSA clustering method, the $C_{\min}$ and $C_{\max}$ are given as 2 and 10, respectively. After clustering, the curve of validity index versus cluster number is shown in Fig. 1, by which the optimal number of clusters is six. Thus, there are six rules generated automatically for the NFS predictor. The settings for PSO and RLSE are given in Table 1 and the parameters of the proposed PSO-RLSE-PSO are shown in Table 2. For the CYN-USD exchange rate forecasting, the NFS predictor with four

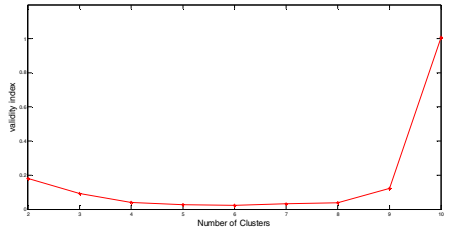


Fig. 1. Validity index versus cluster number by the FBSA clustering method

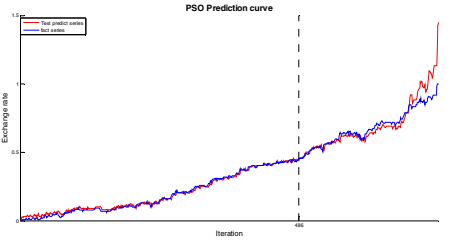


Fig. 2. Predicting response by the NFS predictor with six fuzzy rules (with PSO learning method only)

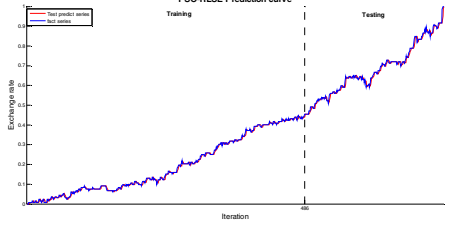


Fig. 3. Predicting response by the NFS predictor with six fuzzy rules (with PSO-RLSE learning method)

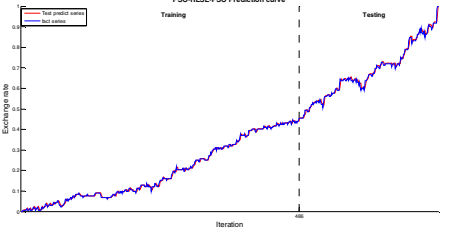


Fig. 4. Predicting response by the NFS predictor with six fuzzy rules (with PSO-RLSE-PSO learning method)

rules is trained by the PSO method only, the PSO-RLSE method, and the PSO-RLSE-PSO method, respectively. Ten trials are conducted for each experiment. The learning iterations for PSO and PSO-RLSE are set to 1000 and 500, respectively. The forecasting accuracy in RMSE for the training and testing are summarized in Table 2 for performance comparison.

Table 2. Parameters after learning by the PSO-RLSE-PSO

Premise-part = x_1 is $s_1(h_1(t))$ and x_2 is $s_2(h_2(t))$				
Parameter	m	σ	m	σ
Rule 1	0.7273	-6.5392	-3.006	-1.0228
Rule 2	2.32606	-9.0054	-4.2367	-1.5436
Rule 3	-9.3967	-3.1629	5.4079	3.5961
Rule 4	-3.4467	1.1636	3.3745	2.8554
Rule 5	-10.5639	2.5563	9.2401	0.3268
Rule 6	7.9862	1.6916	-10.4697	-3.3565
Consequent-part = $a_0+a_1h_1(t)+a_2h_2(t)$				
Parameter	a_0	a_1	a_2	
Rule 1	-0.2478	-0.0592	0.9167	
Rule 2	-0.0371	0.1809	0.6538	
Rule 3	0.5980	0.1124	0.6040	
Rule 4	0.2942	0.5874	0.5094	
Rule 5	-9.0993e-06	-0.0438	-0.0010	
Rule 6	0.9698	0.0017	0.0003	

Table 3. Forecasting performance comparison in RMSE (mean/std)

Method	PSO	PSO-RLSE	PSO-RLSE-PSO
Training phase	0.1036/0.0696	0.0509/0.0484	0.0397/0.0126
Testing phase	0.1434/0.3048	0.0333/0.0316	0.0277/0.0022

Note that above results are based on ten trials.

4 Discussion and Conclusion

The self-organization neural fuzzy approach to the forecasting problem of the CNY-USD exchange rate has been presented. The design of the NFS-based predictor includes the structure-learning (self-organization) phase and the parameter learning phase. In the structure learning phase, there are no fuzzy rules at beginning. The NFS-based predictor is formed by the FBSA clustering method to automatically determine the optimal number of fuzzy If-Then rules and their complexity. After this, the parameter learning phase follows to fine-tune the predictor for good forecasting performance. There are three machine-learning methods used for parameter learning,

Which are the PSO method, the PSO-RLSE method and the PSO-RLSE-PSO method. Although the PSO learning algorithm can adapt the predictor for the problem, the forecasting performance is merely plain. This could be because the parameter space was large, formed by the premise parts and the consequent parts, and the PSO might not easy to find the optimal or near-optimal solution. We further devised the PSO-RLSE hybrid learning method to train the NFS predictor and found the forecasting performance in accuracy got much improved, as shown in Table 3. The forecasting improvement may thank to the divide-and-conquer concept, based on which the PSO-RLSE method is devised. The parameter space was divided into two subspaces, the subspace of the premise parameters and the subspace of the consequent parameters. The PSO and the RLSE were used in hybrid way to search for the optimal solution, as stated previously. The PSO-RLSE method adapted the NFS predictor for forecasting not only in performance improvement but also in fast learning convergence. For this study, we further proposed the PSO-RLSE-PSO method to train the predictor to further improve the forecasting performance, and it worked, as shown in Table 3. The idea for the method is that based on the result by the PSO-RLSE, the PSO is used to further update the consequent parameters for better performance, while the premise parameters are fixed. Although the RLSE is good for linear regression model, it may not be good enough for the nonlinear forecasting problem, such as the CNY-USD exchange rate, which is usually fluctuated constantly and sometimes vibrated radically. This motivates the further use of PSO to the improvement search for the consequent parameters. Thus, the proposed PSO-RLSE-PSO method not only can preserve the merit of the PSO-RLSE method but also can deal with nonlinear forecasting problem with better performance.

The contribution of this research is three-fold. Firstly, the structure of the knowledge base in terms of fuzzy If-Then rules for the NFS-based intelligent predictor is automatically determined by the FBSA clustering method. Secondly, the innovation of the swarm-intelligence-based PSO-RLSE-PSO method is devised and applied successfully to the intelligent predictor for the nonlinear forecasting problem of the CNY-USD exchange rate. Thirdly, through performance comparison, excellent performance by the proposed approach has been shown.

Acknowledgment. This research work is supported by the National Science Council, Taiwan, ROC, under the Grant contract no. NSC99-2221-E-008-088.

References

1. Bordo, M.D., Eichengreen, B.J.: A retrospective on the Bretton Woods system: lessons for international monetary reform. University of Chicago Press, USA (1993)
2. Bodie, Z., Kane, A., Marcus, A.J.: Essentials of Investments. McGraw-Hill/Irwin, Boston (2004)
3. Hong, Y., Lee, T.H.: Inference on predictability of foreign exchange rates via generalized spectrum and nonlinear time series models. The Review of Economics and Statistics 85, 1048–1062 (2003)
4. Molodtsova, T., Papell, D.H.: Out-of-sample exchange rate predictability with Taylor rule fundamentals. Journal of International Economics 77, 167–180 (2009)
5. Leu, Y., Lee, C.P., Jou, Y.Z.: A distance-based fuzzy time series model for exchange rates forecasting. Expert Systems with Applications 36, 8107–8114 (2009)

6. Goldberg, E.: Optimal signal multi-resolution by genetic algorithms to support artificial neural networks for exchange-rate forecasting. Journal of Political Economy 104, 510–541 (1996)
7. Zhang, G., Hu, M.Y.: Neural network forecasting of the British pound/US dollar exchange rate. Omega 26, 495–506 (1998)
8. Jang, S.R.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on System, Man, and Cybernetics 23, 665–685 (1993)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neuro Network Proceedings, vol. 4, pp. 1942–1948 (1995)
10. Chang, J.F., Chu, S.C., Roddick, J.F., Pan, J.S.: A parallel particle swarm optimization algorithm with communication strategies. Journal of Information Science and Engineering 21(4), 809–818 (2005)
11. Huang, H.C., Chen, Y.H., Abraham, A.: Optimized watermarking using swarm-based bacterial foraging. Journal of Information Hiding and Multimedia Signal Processing 1(1), 51–58 (2010)
12. Holland, J.H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor (1975)
13. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
14. Nelder, J.A., Mead, R.: A simplex method for function minimization. The Computer Journal 7, 308–313 (1965)
15. Storey, C.: Applications of a hill climbing method of optimization. Chemical Engineering Science 17, 45–52 (1962)
16. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing: a computational approach to learning and machine intelligence. Prentice Hall, USA (1997)
17. Xiu, Z.H., Ren, G.: Stability analysis and systematic design of Takagi-Sugeno fuzzy control systems. Fuzzy Sets and Systems 151, 119–138 (2005)
18. Sun, H., Wang, S., Jiang, Q.: FCM-based model selection algorithms for determining the number of clusters. Pattern Recognition 37, 2027–2037 (2004)
19. Goodwin, G.C., Sin, K.S.: Adaptive filtering prediction and control. Dover Publications, Inc., New York (2009)
20. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of International Joint Conference on Neural Networks (IJCNN 2004), Budapest, Hungary) (2004)
21. Huang, G.B., Liang, N.Y., Rong, H.J., Saratchandran, P., Sundararajan, N.: On-line sequential extreme learning machine. In: The IASTED International Conference on Computational Intelligence (CI 2005), Calgary, Canada (2005)
22. Nauck, D., Kruse, R.: Neuro-fuzzy systems for function approximation. Fuzzy Sets and Systems 101, 261–271 (1999)
23. Ying, H.: General interval type-2 Mamdani fuzzy systems are universal approximators. In: Fuzzy Information Processing Society, NAFIPS 2008, Annual Meeting of the North American, pp. 1–6 (2008)
24. Li, C., Lee, C.Y., Cheng, K.H.: Pseudo-error-based self-organizing neuro-fuzzy system. IEEE Transactions on Fuzzy Systems 12, 812–819 (2004)
25. Li, C., Lee, C.Y.: Self-organizing neuro-fuzzy system for control of unknown plants. IEEE Transactions on Fuzzy Systems 11, 135–150 (2003)
26. Dunn, J.C.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Cybernetics and Systems 3, 32–57 (1973)
27. Ramze, R.M., Lelieveldt, B.P.F., Reiber, J.H.C.: A new cluster validity index for the fuzzy c-mean. Pattern Recognition Letters 19, 237–246 (1997)
28. OANDA, `http://www.oanda.com/currency/historical-rates`

A Hybrid Model for Credit Evaluation Problem

Hui Fu and Xiaoyong Liu

Department of Computer Science, Guangdong Polytechnic Normal University, Guangzhou, Guangdong, 510665, China
lindafh819@126.com

Abstract. This paper provides a novel hybrid model to solve credit scoring problems. This model is based on RBF neural network with genetic algorithm and its principal character is that Central position, center spread and weights of RBF neural network are encode as genes of Chromosome in genetic algorithm. And then using genetic algorithm trains RBF neural network circularly. A real world credit dataset in the University of California Irvine Machine Learning Repository are selected for the experiment. Numerical experiment shows that the model possesses fast learning ability and excellent generalization ability, and verifies that the novel model has better preference.

Keywords: Credit scoring; RBF neural network; Genetic Algorithm.

1 Introduction

Recently, credit scoring has become widely used in issuing credit cards and in other types of consumer lending, such as auto loans and home equity loans. Credit scoring is a method of evaluating the credit risk of loan applications. The method produces a "score" that a bank can use to rank its loan applicants or borrowers in terms of risk. A better model should give a higher percentage of high scores to borrowers whose loans will perform well and a lesser percentage of low scores to borrowers whose loans won't perform well. Even a good scoring system won't predict with certainty any individual loan's performance, but it should give a fairly accurate prediction of the likelihood that a loan applicant with certain characteristics will default. To build a good scoring model, developers need sufficient historical data, which reflect loan performance in periods of both good and bad economic conditions.[1][2]

There are mainly two types of credit scoring models, statistical methods and machine learning methods. Several statistical methods, such as linear probability models, logistic models and probabilistic models, are used to develop credit scoring systems. They are standard statistical techniques for estimating the probability based on historical data on loan performance and characteristics of the borrower [1]. The second method is based on machine learning (examples of machine learning techniques used to solve the above financial decision-making problems are Atiya[3]; Huang, Chen, Hsu, Chen, & Wu[4]; Lee, Chiu, Chou, & Lu[5]) use the multi-layer perceptron (MLP) as classifier. Other tested classifiers are the Decision Tree and the Support Vector Machine [6-8]. We want to stress that these studies show that the machine learning based systems are better than the traditional (statistical) methods for bankruptcy

Y. Tan et al. (Eds.): ICSI 2011, Part I, LNCS 6728, pp. 626–634, 2011.
© Springer-Verlag Berlin Heidelberg 2011

prediction and credit scoring problems (Huang et al.[4]; Ong, Huang, & Tzeng[9]; Vellido, Lisboa, & Vaughan[10]; Wong & Selvi[11]). In Tsai and Wu[12] the authors compare a single MLP classifier with multiple classifiers and diversified multiple classifiers on three datasets. Artificial neural network models are used to solve credit evaluation problem. However, neural network has some shortcomings, such as slow training speed and local optimum. This paper presents a new model, which combines artificial neural network and genetic algorithm and trains neural network circularly using genetic algorithm.

2 RBF Neural Network and Genetic Algorithm

2.1 RBF Neural Network

Radial basis function neural network is different from classic BP neural network. Design of RBF is seen as a curve fitting problem in high-dimensional space, so the network learning is equivalent to the multidimensional space to find a surface to fit the given dataset. Although theory of RBF neural network and BP neural network is different, RBF neural network, like BP neural network, can approximate any nonlinear functions. In addition, RBF neural network is better than BP neural network in the generalization ability. And more, RBF neural network converges faster than BP neural network. From network structure, RBF neural network has also three layers, the input layer, hidden layer and output layer, but hidden layer has only one layer, while the BP neural network have two or more layers, The structure of RBF neural network is shown in Fig.1 as follow.

The output of RBF neural network is according to expression as follows:

$$y = f(x) = \sum_{k=1}^{n} w_k \cdot \varphi(x, c_k) = \sum_{k=1}^{n} w_k \cdot \varphi(\|x - c_k\|) \tag{1}$$

Where,

x is an input vector,

φ is a processing function for the hidden layer. In this paper, φ is the Gaussian function .

$\| \ \|$ denotes the Euclidean norm,

w_k is weight value of the output layer weights,

n is the number of neurons in hidden layer,

c_k is RBF centers.

Different with BP, neurons in hidden layer of RBF neural network is the RBF center, whose role is to calculate the distance between the center and the network input, and then transmit to output layer for processing. RBF output function, φ ,is defined as follows:

$$\varphi(x, t_k) = \exp\left(- \frac{L}{d_{\max}^2 \|x - t_k\|^2} \right) \tag{2}$$

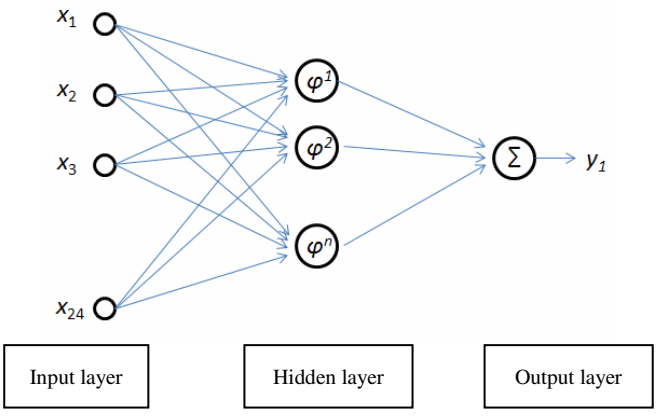


Fig. 1. The structure of RBF neural network

2.2 Genetic Algorithm

Genetic algorithm (GA) is an adaptive global optimization statistical search algorithm, which simulate biological evolution process in natural environment. GA was provided by Professor *Holland* in 1970's, and then, *De Jong* carried out several numerical experiments to verify its preference. *Goldberg* summarized related study and formed the basic framework of genetic algorithm in 1980's. Based on some common features, Goldberg summarized a basic genetic algorithm (Simple Genetic Algorithms, SGA for short). Simple genetic algorithm included three basic genetic operators, selection, crossover and mutation.

Select operator can exclude worst individuals, and improve the global convergence and computational efficiency. In biological evolution process, crossover is a major component and can produce new individuals or species. Genetic algorithm uses this feature to produce a new generation with character of parents individual. Mutation operator can generate individuals with new features.

3 GA-RBF Neural Network Model

This section will describe the process that using genetic algorithm and RBF neural network construct personal credit assessment model. This model applies genetic algorithm to establish the overall framework and merges RBF neural networks to the framework. Central position, center spread and weights of RBF neural network can be improved iteratively and increase predictive rate. In each iterations, there are three major steps, select operation, crossover operation and mutation operation. Select operation will retain excellent individuals and eliminate negative individuals. Crossover operation and mutation operation are able to reproduce a new individual. This model process is in detail as follows:

Step 1:

Parameter settings.

Setting the population size *M,* crossover probability $P_{c,}$ mutation probability $P_{m,}$.

Step 2:

Set the fitness function.

Fitness function is the optimization goal, where is denoted by predictive rate.

Step 3 :

Population initialization.

Each individuals within the population *M* is initialized, viz, establish *M* different RBF neural network model.

Step 4:

Individual assessment.

Verify whether all of the individual to achieve the desired results according to the group fitness function, and if so, input the best individual parameters.

Step 5:

Select operation.

Calculate all the individuals' fitness $F_{i,}$ in population M. F_i is predictive rate for the unknown data.

Select probability, p_i, can be defined as follows,

$$p_i = \frac{F_i}{\sum_{i=1}^{M} F_i} \qquad i = 1,2,3,\cdots M \tag{3}$$

According p_i, calculate the next generation population.

Step 6:

Chromosome encoding.

Weights of each individual correspond to genes of a chromosome, as shown in Fig.2. Population will produce M chromosomes.

Step 7:

Crossover operation.

The group of M chromosomes make pairs randomly and form M / 2 pairs of chromosomes for crossover operation. This paper uses double-point crossover.

Step 8:

Mutation operation.

Each gene of chromosomes can mutate randomly according to the probability P_m.

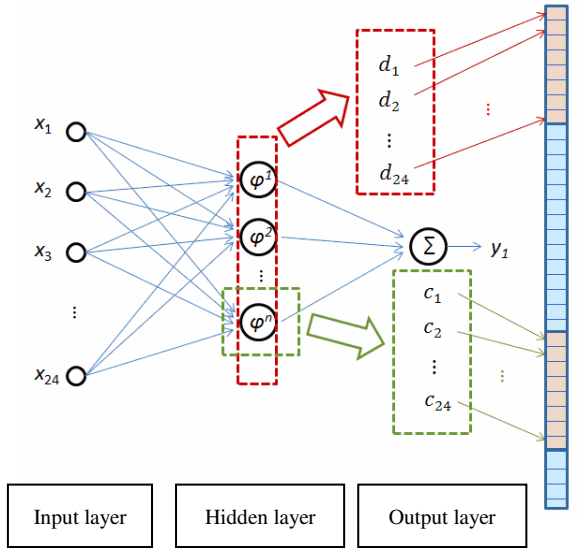


Fig. 2. Central position, center spread and weights of RBF neural network and weights transfer Chromosome encoding

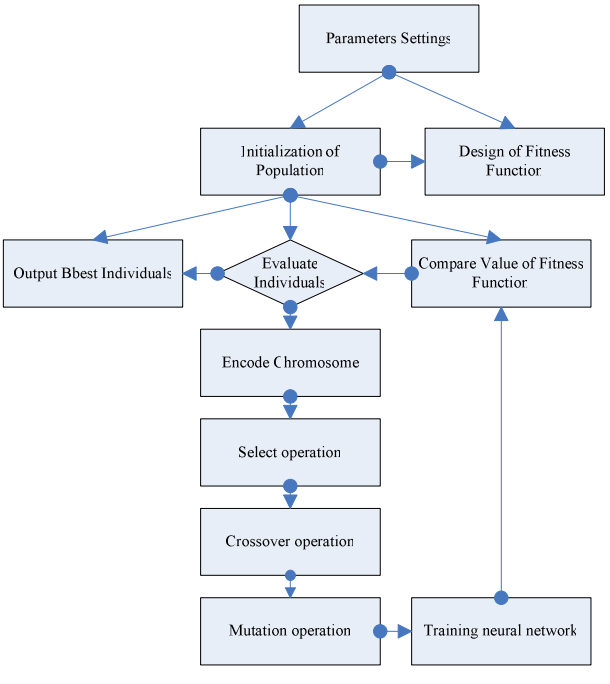


Fig. 3. The flow chart of GA-RBFNN

Step 9:
Training neural network.

The new generation of chromosomes correspond to the RBF neural network and training it.

Step 10:
Recalculate fitness values of *M* individuals and go back to step 4.

The flow chart of GA-RBFNN is shown in fig.3.

4 GA-RBF Neural Network Model to Evaluate the Application of Personal Credit

4.1 Dataset and Pretreatment

Test dataset is personal credit assessment data of Germany Bank from the University of California Irvine Machine Learning Repository. German credit card dataset has 1000 Instances. There are twenty-four numerical attributes in this dataset. Number of customers as "good" is 700, and number of customers as "bad" is 300. In the dataset, "0" denotes a customer as "bad", and "1" denotes a customer as "good". 80% of the dataset is used for training, and the rest is used for validating.

In German credit card dataset, data of different attribute has different range. The largest range is between 0 and 184, while the smallest range is 0 to 1. Because learning rate of the weights in neural network is uniform, it will be instability for data of attribute with larger range, and simultaneously speed lower time to training for data of attribute with smaller range. In a word, it is necessary to implement the normalization operation as preprocessing before training.

In this paper, the goal of pretreatment is that data of different attribute has same range, which is 0 to 1. Pretreatment process is denoted for normalized as follows:

Step 1:
Calculate maximum and minimum values with the same attribute.

Step 2:
Normalize the given data with the same attribute according to expression as follow,

$$D_i = \frac{d_i - d_{\min}}{d_{\max} - d_{\min}} \qquad i = 1,2,3,\cdots,m \tag{4}$$

Where,

D_i is data that has been processed,

d_i is data that will be processed,

d_{max} is the maximum value in the given attribute,

d_{min} is the minimum value in the given attribute.

4.2 Parameters Setting and Experiment Results

After the preliminary design model, this paper uses Visual Studio 2005 as development platform. Related parameters are set as follows:

Number of neurons in Input layer : 24
Number of neurons in Hidden layer: 10
Learning rate of weights: 0.02
Learning rate of Center position: 0.01
Learning rate of Center spread: 0.03
Population size: M = 30
Crossover probability: 0.002
Mutation probability: 0.001

The prediction rate is shown in fig.4.

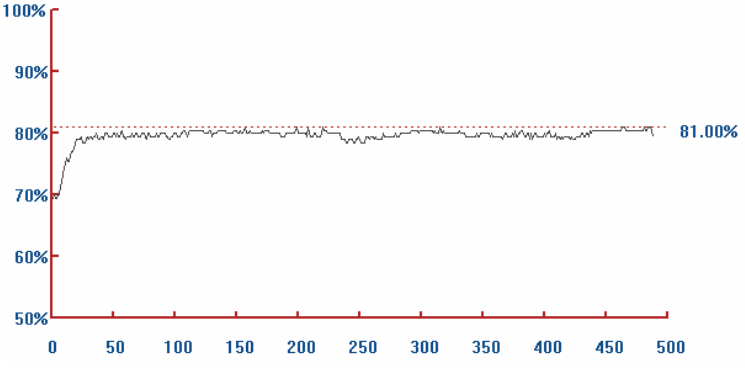


Fig. 4. The Change of prediction rate for personal credit data by GA-RBF

Prediction rate of GA-RBF can achieve 81.0 % at best.
The same parameter settings compared RBF neural networks:
Number of neurons in Input layer : 24
Number of neurons in Hidden layer: 10
Learning rate of weights: 0.02
Learning rate of Center position: 0.01
Learning rate of Center spread: 0.03

The prediction rate is shown as follows.

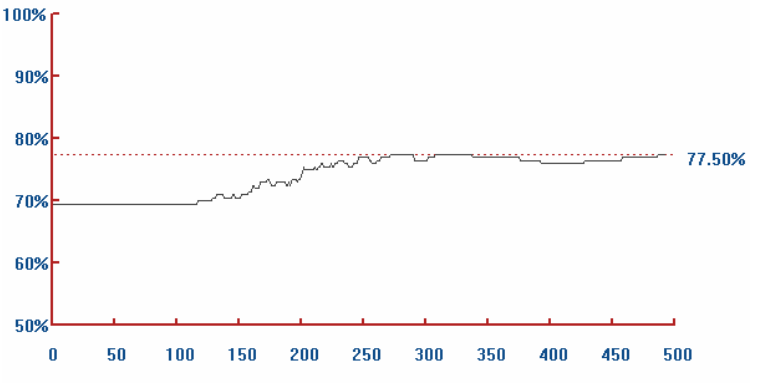


Fig. 5. The Change of prediction rate for personal credit data by RBF

It's obviously seen that GA-RBF can get higher prediction rate than RBF, increase about 5%.

5 Conclusion

This paper constructs a new hybrid personal credit scoring model by combining RBF neural network with genetic algorithm. The new model uses genetic algorithm to optimize weights, the centers position and the centers spread of RBF neural network, so that convergence of the model is rapid and has excellent generalization in numerical experiments.

Acknowledgement

The author would like to thank anonymous reviewers for their constructive and enlightening comments, which improved the manuscript. This work has been supported by grants from Program for Excellent and Creative Young Talents in Universities of Guangdong Province (LYM09097 and LYM10097), and Guangdong Polytechnic Normal University (No.10kjy02).

References

1. McAllister, P.H., Mingo, J.J.: Commercial Loan Risk Management, Credit-Scoring, and Pricing: The Need for a New Shared Database. Journal of Commercial Lending, 6–22 (1994)
2. Mester, L.: What's the point of credit scoring? Business review (3), 3–16 (1997)

3. Atiya, A.F.: Bankruptcy prediction for credit risk using neural networks: a survey and new results. IEEE Transactions on Neural Networks 12(4), 929–935 (2001)
4. Huang, Z., Chen, H., Hsu, C.-J., Chen, W.-H., Wu, S.: Credit rating analysis with support vector machines and neural networks: A market comparative study. Decision Support Systems (37), 543–558 (2004)
5. Lee, T.-S., Chiu, C.-C., Chou, Y.-C., Lu, C.-J.: Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. Computational Statistics and Data Analysis (50), 1113–1130 (2006)
6. Stecking, R., Schebesch, K.B.: Combining Support Vector Machines for Credit Scoring. In: Proceedings of Operations Research, pp. 135–140 (2006)
7. Martens, D., Baesens, B., Van Gestel, T., et al.: Comprehensible credit scoring models using rule extraction from support vector machines. European Journal of Operational Research 183(3), 1466–1476 (2007)
8. Luo, S., Cheng, B., Hsieh, C.: Prediction model building with clustering-launched classification and support vector machines in credit scoring. Expert Systems with Applications 36(4), 7562–7566 (2009)
9. Ong, C.-S., Huang, J.-J., Tzeng, G.-H.: Building credit scoring models using genetic programming. Expert Systems with Applications (29), 41–47 (2005)
10. Vellido, A., Lisboa, P.J.G., Vaughan, J.: Neural networks in business: a survey of applications (1992–1998). Expert Systems with Applications 17, 51–70 (1999)
11. Wong, B.K., Selvi, Y.: Neural network applications in finance: A review and analysis of literature (1990–1996). Information and Management 34, 129–139 (1998)
12. Tsai, C., Wu, J.: Using neural network ensembles for bankruptcy prediction and credit scoring. Expert Systems with Applications 34(4), 2639–2649 (2008)

Author Index

Adams, Martin II-519
Aiping, Zhang I-347
Amin-Naseri, Mohammad Reza II-252
An, Jinung II-91, II-99
Ariffin, Junaidah I-182

Bai, Jie I-64, I-355
Bastos-Filho, Carmelo José Albanez II-543, II-563
Batouche, Mohamed I-221
Belaton, Bahari II-173
Ben-Abdallah, Hanêne I-250
Bilal, Mohsin I-530
Breedon, Philip I-19

Castro, Rodrigo M.C.S. II-543
Cavalcanti-Júnior, George M. II-543
Chai, Yi I-260
Chao, Chih-Chiang II-66
Chen, Chien-Hsing II-236, II-269
Chen, JenLian I-28
Chen, Lei II-411
Chen, Min-you I-165
Chen, Powen I-56
Chen, Qinglan II-502
Chen, Walter I-56
Chen, Wei II-465
Chen, Weili II-164
Chen, Weirong I-310, I-338
Chen, Xue-bo II-82, II-449
Chen, Xue-jun II-441
Chen, Yanju I-329
Chen, Yingge II-317
Chen, Zhenmin II-434
Cheng, Shi I-38
Chhabra, Jitender Kumar I-147, II-146
Chiang, Ya-Tzu II-66
Choi, Kyung-Sik II-91
Chou, PenChen I-28
Chunguo, Wu II-275
Chuyi, Song II-275
Coelho, André L.V. II-573

Dahiya, Surender Singh I-147
Dai, Chaohua I-310
De Giusti, Armando I-111
de Lima-Neto, Fernando Buarque II-543, II-563
Deng, Changshou I-416
Deng, Zhi I-277, I-285
Dieck Kattas, Graciano I-9
Di-Ming, Ai II-10
Ding, Yongsheng I-130
Djerou, Leila I-221
Du, Yu I-310
Duan, Pengfei I-234, II-200

Elek, István II-291, II-299
Elkamel, Akil I-250

Fang, Gang II-157
Fang, Lijing II-457
Fei, MinRui I-93, II-74
Fekete, István II-299
Feng, Haizhou I-285
Feng, Pan II-10
Feng, Yuanjing I-267
Fernández-Martínez, Juan Luis I-1
Ferreira, Cláudio I-595
Fu, Bo II-327
Fu, Hui I-626
Fu, Wei I-464
Fu, Xiping I-93, II-41

Gao, Liang II-1
Gao, Yuelin I-101
García-Gonzalo, Esperanza I-1
Garro, Beatriz A. I-242, I-447
Ge, Hong I-408
Ghedira, Khaled II-283
Gin, Yue I-347
Grissa Touzi, Amel II-191
Gu, Huaxi I-277, I-285
Gu, Yu II-529
Guang, Ren I-347
Guo, Fenhong II-395
Guo, Weidong II-411
Guo, Yun-fei II-419
Gzara, Mariem I-250, II-136

Habibnejad Korayem, Moharam I-606
Han, Fengling II-348
Han, Peng I-86
He, Wei I-165, II-441
He, Zhen I-539
Holban, Ştefan I-46
Hoseinnezhad, Reza II-509
Hoseinpour, Farhoud I-606
Hoseinzadeh, Mojtaba I-606
Hou, Guolian II-379
Hsu, Chung-Chian II-236
Hu, Gang I-165
Huang, Hongming I-616
Huang, Li II-340
Huang, Tian-yun II-82
Huang, Yizhou II-118

jalali, Mohammad jafarpour II-371
Janecek, Andreas II-307, II-553
Jernigan, Robert I-1
Ji, Zhen I-587
Jia, Liyuan II-340
Jia, Yutian I-172
Jianda, Han II-108
Jiang, Shouda I-400
Jin, Linpeng I-374
Jingqing, Jiang II-275

Kala, Rahul I-480
Khan, Farrukh Aslam I-530
Khan, Salabat I-530
Khelil, Naceur I-221
Khereddine, Besma II-136
Kim, Kyoung-Dong II-99
Kim, Yoon-Gu II-91, II-99
King, David I-19
Kloczkowski, Andrzej I-1
Kou, Jialiang I-234, I-431, II-200
Kulworawanichpong, Thanatchai II-403
Kumar, Shakti I-147
Kwon, Jaerock I-203

Lai, Mao-sheng II-419
Lambert-Torres, Germano I-595
Lanzarini, Laura I-111
Lee, Suk-Gyu II-91, II-99
Leeton, Uthen II-403
Lei, Fanfan I-101
Leow, Amy II-173
Li, Bin I-539
Li, Chaoyang II-474
Li, Chunshien I-616
Li, Fengjun I-472
Li, Hailing II-457
Li, Haiyuan II-118
Li, Hui I-234, I-424, II-200
Li, Huirong I-101
Li, Jianbiao II-474
Li, Jie I-260
Li, Jimin I-101
Li, Junli I-374
Li, Lei II-340
Li, Lingyan II-465
Li, Nan I-506
Li, Ning II-434
Li, Peigen II-1
Li, Penghua I-260
Li, Wenhui II-327
Li, Wenqi I-194, I-227
Li, Xiaodong I-497
Li, Ying II-379
Liang, Xinyuan I-157
Liao, Wudai I-80
Lihu, Andrei I-46
Lin, Chuan Wei I-616
Lin, Dongmei I-497
Lin, Feng II-419
Lin, Kunming II-348
Lin, Yifeng II-327
Lin, Yun II-479
Ling, Haifeng II-57
Liu, Guoquan I-464
Liu, Hongbing I-234, II-363
Liu, Jian II-183
Liu, Jianming II-434
Liu, Ju II-411
Liu, Qun II-17, II-207
Liu, Susu II-356
Liu, Tingzhan II-183
Liu, Weifeng II-486
Liu, Wenmin I-587
Liu, Xiangkai II-387
Liu, Xiaoni I-578
Liu, Xiaoyong I-626
Liu, Yankui I-559, II-164
Liu, Yi I-234, I-424, I-431, II-200
Liu, Ying I-329
Liu, Yuxi II-411
López, Javier I-111
Lou, Yang I-374

Lu, Xinzhong II-427
Lu, Yinan I-578
Luo, Yi I-506

Madeiro, Salomão Sampaio II-563
Mansouri, Mohammad I-321
Mao, Yunfei II-74
Mao, Yu-xing II-441
Márton, Mátyás II-291
Maulini, Juan Andrés I-111
Menezes, Lara II-573
Meng, Jixian II-427
Menhas, Muhammad Ilyas II-41
Menhas, Muhammd Ilyas I-93
Mesghouni, Nadia II-283
Meybodi, Mohammad Reza I-120
Mo, Hongwei I-390, I-400
Mohamed, Azlinah I-182
Mullane, John II-519

Naebi, Ahmad I-120, I-606
Nascimento Figueiredo, Elliackin Messias do II-563
Neshat, Najmeh II-252
Nie, Li II-1
Ningning, Liu II-34
Niu, Qun II-74

Orouskhani, Maysam I-321
Ouyang, Xin-yu II-449

Paes Salomon, Camila I-595
Pan, Chang chun I-64, I-355
Pang, Shanchen II-228
Parashar, Anshu II-146
Peng, Dong-liang II-529
Peng, Hu I-416
Pereira Coutinho, Maurilio I-595

Qi, Jie I-130
Qian, Tao I-364
Qiu, Yiming I-194, I-227
Qu, Zheng I-390
Quan, Yong I-578

Rabevohitra, Feno Heriniaina I-212
Ramadass, Sureswaran I-606
Ran, Lili I-310
Ren, Xue II-474
rezaee, Alireza II-371
Rezazadeh, Iman I-120
Rui, Zhang II-10

Sang, Jun I-212
Saraswathi, Saras I-1
Sassi, Minyar II-191
Sharif, Muhammad I-530
Shen, Gang I-522
Shen, Yuanxia II-17
Shi, Yuhui I-38, I-71, I-303, I-374
Shu, Baojian I-285
Shukla, Anupam I-480
Small, Michael I-9
Song, Yong Duan I-86
Sossa, Humberto I-447
Sun, Limin II-356
Sun, Mingfang II-333
Sun, Zhibin II-395
Szendrei, Rudolf II-291, II-299

Tahooneh, Amin I-293
Tajudin Khader, Ahamad II-173
Tan, Jindong II-118
Tan, Ying I-382, I-549, II-218, II-307, II-553
Tang, Ke Ming II-260
Tang, Lixin II-26
Tang, Shi Xi II-260
Tang, Xu-Qing I-568
Tarsauliya, Anupam I-480
Temani, Moncef II-283
Teshnehlab, Mohammad I-321
Tewolde, Girma S. I-203
Thabet, Aicha II-191
Thomas, J. Joshua II-173
Tiwari, Ritu I-480

Vázquez, Roberto A. I-242, I-447
Vo, Ba-Ngu II-509, II-519
Vo, Ba-Tuong II-519
Vu, Truong Nguyen II-509

Wan, Chunqiu I-439
Wan, Shuzhen I-234, I-424, I-431
Wang, Chao I-157
Wang, Dong I-497
Wang, Fei II-537
Wang, Guoyin II-17
Wang, Jiangfeng I-80
Wang, Jilin II-395

Wang, Jiye II-434
Wang, Jun I-439, I-549
Wang, Junhui I-277
Wang, Junyan I-80
Wang, Kebin II-218
Wang, Lei I-138, I-194, I-227
Wang, Lihe I-455
Wang, Ling I-93, II-41, II-74
Wang, Qiquan II-537
Wang, Sujing II-333
Wang, Wei II-82
Wang, Xianpeng II-26
Wang, Xiaojuan II-1
Wang, Xiaoqing I-559
Wang, Xiaoying II-317
Wang, Xin I-157
Wang, Xu I-489, II-479
Wang, Yu I-539
Wang, Zhejin I-267
Wei, Hongxing II-118
Wei, Qiming I-416
Wen, Chenglin II-486
Wu, Chang-an II-363
Wu, Jianping I-172
Wu, Lenan I-514
Wu, Qi di I-138, I-194, I-227
Wu, Quanjun II-128
Wu, Tao II-207
Wu, Xiaoli II-164
Wu, Yali II-49
Wu, Yingbo I-489, II-479
Wu, Yuheng I-364

Xian, Xingping II-207
Xiao, Yu II-387
Xiaowei, Qi I-347
Xiong, Jiang II-157
Xiong, Shengwu I-234, I-424, I-431, II-200
Xiong, Yan II-363
Xu, Baoguo II-494
Xu, Benlian II-494, II-502, II-537
Xu, Li II-128
Xu, Lifang I-400
Xu, Liqing II-49
Xu, Qingshan I-157
Xu, Sai II-474
Xu, Wang-bao II-82
Xu, Yanxin I-64
Xu, Yayi II-34
Xu, Zhi-Guang II-91, II-99
Xue, Jingqian II-49

Yan, XueMing I-408
Yanchun, Liang II-275
Yang, Dongsheng I-227
Yang, Geng I-439
Yang, Gen Ke I-64, I-355, I-455
Yang, Liying II-244
Yang, Shenghui II-387
Yang, Simon X. I-464
Yang, Xiaoyun II-387
Yang, Yanling I-416
Yang, Yintang I-277
Yang, Yong-ming II-348, II-441
Yao, Gang I-157
Ye, Wei II-41
Yeung, Lam Fat I-455
Yin, Hongpeng I-260
Yin, Peng-Yeng II-66
Yin, Zhe II-419
Ying, Hong II-157
Yu, Zhonglei II-228
Yuqing, He II-108
Yusoff, Marina I-182

Zhai, Jin-qian I-165
Zhang, Hong II-465
Zhang, Hua II-128
Zhang, Jianhong II-395
Zhang, Jianhua II-379
Zhang, Jie II-465
Zhang, Kaibo I-539
Zhang, Kun I-568
Zhang, Liwei I-559
Zhang, Lu I-86
Zhang, Na II-333
Zhang, Wenfang II-379
Zhang, Xing I-439
Zhang, Xuexia I-338
Zhang, Yan-Qing I-522
Zhang, Yi I-310
Zhang, Yongguo II-34
Zhang, Yong-wei I-138
Zhang, Yudong I-514
Zhang, Zili I-364
Zhang, Ziqiu II-57
Zhang, Zulong II-348
Zhao, Bingyan I-416
Zhao, Jing II-457

Zhao, Ling II-183
Zhao, Yan I-578
Zhao, Yong-jian II-157
Zhe, Zhang II-10
Zheng, Dong II-411
Zheng, Mingfa II-34
Zheng, Wang II-108
Zheng, Yihui I-157
Zheng, Yujun II-57
Zhou, Chunguang II-333
Zhou, Huan I-364
Zhou, Jin II-128
Zhou, Lidan I-157
Zhou, Wenyong II-363
Zhou, Xianzhong II-57
Zhou, Zheng I-71
Zhu, Jihong II-502, II-537
Zhu, Jun I-355
Zhu, Peiyi II-494
Zhu, Yuanchun I-382
Zhu, Zexuan I-587
Ziarati, Koorush I-293
Zong, Xinlu I-234
Zuo, Xingquan I-172

Author Index

Adams, Martin II-519
Aiping, Zhang I-347
Amin-Naseri, Mohammad Reza II-252
An, Jinung II-91, II-99
Ariffin, Junaidah I-182

Bai, Jie I-64, I-355
Bastos-Filho, Carmelo José Albanez II-543, II-563
Batouche, Mohamed I-221
Belaton, Bahari II-173
Ben-Abdallah, Hanêne I-250
Bilal, Mohsin I-530
Breedon, Philip I-19

Castro, Rodrigo M.C.S. II-543
Cavalcanti-Júnior, George M. II-543
Chai, Yi I-260
Chao, Chih-Chiang II-66
Chen, Chien-Hsing II-236, II-269
Chen, JenLian I-28
Chen, Lei II-411
Chen, Min-you I-165
Chen, Powen I-56
Chen, Qinglan II-502
Chen, Walter I-56
Chen, Wei II-465
Chen, Weili II-164
Chen, Weirong I-310, I-338
Chen, Xue-bo II-82, II-449
Chen, Xue-jun II-441
Chen, Yanju I-329
Chen, Yingge II-317
Chen, Zhenmin II-434
Cheng, Shi I-38
Chhabra, Jitender Kumar I-147, II-146
Chiang, Ya-Tzu II-66
Choi, Kyung-Sik II-91
Chou, PenChen I-28
Chunguo, Wu II-275
Chuyi, Song II-275
Coelho, André L.V. II-573

Dahiya, Surender Singh I-147
Dai, Chaohua I-310
De Giusti, Armando I-111
de Lima-Neto, Fernando Buarque II-543, II-563
Deng, Changshou I-416
Deng, Zhi I-277, I-285
Dieck Kattas, Graciano I-9
Di-Ming, Ai II-10
Ding, Yongsheng I-130
Djerou, Leila I-221
Du, Yu I-310
Duan, Pengfei I-234, II-200

Elek, István II-291, II-299
Elkamel, Akil I-250

Fang, Gang II-157
Fang, Lijing II-457
Fei, MinRui I-93, II-74
Fekete, István II-299
Feng, Haizhou I-285
Feng, Pan II-10
Feng, Yuanjing I-267
Fernández-Martínez, Juan Luis I-1
Ferreira, Cláudio I-595
Fu, Bo II-327
Fu, Hui I-626
Fu, Wei I-464
Fu, Xiping I-93, II-41

Gao, Liang II-1
Gao, Yuelin I-101
García-Gonzalo, Esperanza I-1
Garro, Beatriz A. I-242, I-447
Ge, Hong I-408
Ghedira, Khaled II-283
Gin, Yue I-347
Grissa Touzi, Amel II-191
Gu, Huaxi I-277, I-285
Gu, Yu II-529
Guang, Ren I-347
Guo, Fenhong II-395
Guo, Weidong II-411
Guo, Yun-fei II-419
Gzara, Mariem I-250, II-136

Habibnejad Korayem, Moharam I-606
Han, Fengling II-348
Han, Peng I-86
He, Wei I-165, II-441
He, Zhen I-539
Holban, Ştefan I-46
Hoseinnezhad, Reza II-509
Hoseinpour, Farhoud I-606
Hoseinzadeh, Mojtaba I-606
Hou, Guolian II-379
Hsu, Chung-Chian II-236
Hu, Gang I-165
Huang, Hongming I-616
Huang, Li II-340
Huang, Tian-yun II-82
Huang, Yizhou II-118

jalali, Mohammad jafarpour II-371
Janecek, Andreas II-307, II-553
Jernigan, Robert I-1
Ji, Zhen I-587
Jia, Liyuan II-340
Jia, Yutian I-172
Jianda, Han II-108
Jiang, Shouda I-400
Jin, Linpeng I-374
Jingqing, Jiang II-275

Kala, Rahul I-480
Khan, Farrukh Aslam I-530
Khan, Salabat I-530
Khelil, Naceur I-221
Khereddine, Besma II-136
Kim, Kyoung-Dong II-99
Kim, Yoon-Gu II-91, II-99
King, David I-19
Kloczkowski, Andrzej I-1
Kou, Jialiang I-234, I-431, II-200
Kulworawanichpong, Thanatchai II-403
Kumar, Shakti I-147
Kwon, Jaerock I-203

Lai, Mao-sheng II-419
Lambert-Torres, Germano I-595
Lanzarini, Laura I-111
Lee, Suk-Gyu II-91, II-99
Leeton, Uthen II-403
Lei, Fanfan I-101
Leow, Amy II-173
Li, Bin I-539
Li, Chaoyang II-474
Li, Chunshien I-616
Li, Fengjun I-472
Li, Hailing II-457
Li, Haiyuan II-118
Li, Hui I-234, I-424, II-200
Li, Huirong I-101
Li, Jianbiao II-474
Li, Jie I-260
Li, Jimin I-101
Li, Junli I-374
Li, Lei II-340
Li, Lingyan II-465
Li, Nan I-506
Li, Ning II-434
Li, Peigen II-1
Li, Penghua I-260
Li, Wenhui II-327
Li, Wenqi I-194, I-227
Li, Xiaodong I-497
Li, Ying II-379
Liang, Xinyuan I-157
Liao, Wudai I-80
Lihu, Andrei I-46
Lin, Chuan Wei I-616
Lin, Dongmei I-497
Lin, Feng II-419
Lin, Kunming II-348
Lin, Yifeng II-327
Lin, Yun II-479
Ling, Haifeng II-57
Liu, Guoquan I-464
Liu, Hongbing I-234, II-363
Liu, Jian II-183
Liu, Jianming II-434
Liu, Ju II-411
Liu, Qun II-17, II-207
Liu, Susu II-356
Liu, Tingzhan II-183
Liu, Weifeng II-486
Liu, Wenmin I-587
Liu, Xiangkai II-387
Liu, Xiaoni I-578
Liu, Xiaoyong I-626
Liu, Yankui I-559, II-164
Liu, Yi I-234, I-424, I-431, II-200
Liu, Ying I-329
Liu, Yuxi II-411
López, Javier I-111
Lou, Yang I-374

Lu, Xinzhong II-427
Lu, Yinan I-578
Luo, Yi I-506

Madeiro, Salomão Sampaio II-563
Mansouri, Mohammad I-321
Mao, Yunfei II-74
Mao, Yu-xing II-441
Márton, Mátyás II-291
Maulini, Juan Andrés I-111
Menezes, Lara II-573
Meng, Jixian II-427
Menhas, Muhammad Ilyas II-41
Menhas, Muhammd Ilyas I-93
Mesghouni, Nadia II-283
Meybodi, Mohammad Reza I-120
Mo, Hongwei I-390, I-400
Mohamed, Azlinah I-182
Mullane, John II-519

Naebi, Ahmad I-120, I-606
Nascimento Figueiredo, Elliackin Messias do II-563
Neshat, Najmeh II-252
Nie, Li II-1
Ningning, Liu II-34
Niu, Qun II-74

Orouskhani, Maysam I-321
Ouyang, Xin-yu II-449

Paes Salomon, Camila I-595
Pan, Chang chun I-64, I-355
Pang, Shanchen II-228
Parashar, Anshu II-146
Peng, Dong-liang II-529
Peng, Hu I-416
Pereira Coutinho, Maurilio I-595

Qi, Jie I-130
Qian, Tao I-364
Qiu, Yiming I-194, I-227
Qu, Zheng I-390
Quan, Yong I-578

Rabevohitra, Feno Heriniaina I-212
Ramadass, Sureswaran I-606
Ran, Lili I-310
Ren, Xue II-474
rezaee, Alireza II-371
Rezazadeh, Iman I-120
Rui, Zhang II-10

Sang, Jun I-212
Saraswathi, Saras I-1
Sassi, Minyar II-191
Sharif, Muhammad I-530
Shen, Gang I-522
Shen, Yuanxia II-17
Shi, Yuhui I-38, I-71, I-303, I-374
Shu, Baojian I-285
Shukla, Anupam I-480
Small, Michael I-9
Song, Yong Duan I-86
Sossa, Humberto I-447
Sun, Limin II-356
Sun, Mingfang II-333
Sun, Zhibin II-395
Szendrei, Rudolf II-291, II-299

Tahooneh, Amin I-293
Tajudin Khader, Ahamad II-173
Tan, Jindong II-118
Tan, Ying I-382, I-549, II-218, II-307, II-553
Tang, Ke Ming II-260
Tang, Lixin II-26
Tang, Shi Xi II-260
Tang, Xu-Qing I-568
Tarsauliya, Anupam I-480
Temani, Moncef II-283
Teshnehlab, Mohammad I-321
Tewolde, Girma S. I-203
Thabet, Aicha II-191
Thomas, J. Joshua II-173
Tiwari, Ritu I-480

Vázquez, Roberto A. I-242, I-447
Vo, Ba-Ngu II-509, II-519
Vo, Ba-Tuong II-519
Vu, Truong Nguyen II-509

Wan, Chunqiu I-439
Wan, Shuzhen I-234, I-424, I-431
Wang, Chao I-157
Wang, Dong I-497
Wang, Fei II-537
Wang, Guoyin II-17
Wang, Jiangfeng I-80
Wang, Jilin II-395

Wang, Jiye II-434
Wang, Jun I-439, I-549
Wang, Junhui I-277
Wang, Junyan I-80
Wang, Kebin II-218
Wang, Lei I-138, I-194, I-227
Wang, Lihe I-455
Wang, Ling I-93, II-41, II-74
Wang, Qiquan II-537
Wang, Sujing II-333
Wang, Wei II-82
Wang, Xianpeng II-26
Wang, Xiaojuan II-1
Wang, Xiaoqing I-559
Wang, Xiaoying II-317
Wang, Xin I-157
Wang, Xu I-489, II-479
Wang, Yu I-539
Wang, Zhejin I-267
Wei, Hongxing II-118
Wei, Qiming I-416
Wen, Chenglin II-486
Wu, Chang-an II-363
Wu, Jianping I-172
Wu, Lenan I-514
Wu, Qi di I-138, I-194, I-227
Wu, Quanjun II-128
Wu, Tao II-207
Wu, Xiaoli II-164
Wu, Yali II-49
Wu, Yingbo I-489, II-479
Wu, Yuheng I-364

Xian, Xingping II-207
Xiao, Yu II-387
Xiaowei, Qi I-347
Xiong, Jiang II-157
Xiong, Shengwu I-234, I-424, I-431, II-200
Xiong, Yan II-363
Xu, Baoguo II-494
Xu, Benlian II-494, II-502, II-537
Xu, Li II-128
Xu, Lifang I-400
Xu, Liqing II-49
Xu, Qingshan I-157
Xu, Sai II-474
Xu, Wang-bao II-82
Xu, Yanxin I-64
Xu, Yayi II-34
Xu, Zhi-Guang II-91, II-99
Xue, Jingqian II-49

Yan, XueMing I-408
Yanchun, Liang II-275
Yang, Dongsheng I-227
Yang, Geng I-439
Yang, Gen Ke I-64, I-355, I-455
Yang, Liying II-244
Yang, Shenghui II-387
Yang, Simon X. I-464
Yang, Xiaoyun II-387
Yang, Yanling I-416
Yang, Yintang I-277
Yang, Yong-ming II-348, II-441
Yao, Gang I-157
Ye, Wei II-41
Yeung, Lam Fat I-455
Yin, Hongpeng I-260
Yin, Peng-Yeng II-66
Yin, Zhe II-419
Ying, Hong II-157
Yu, Zhonglei II-228
Yuqing, He II-108
Yusoff, Marina I-182

Zhai, Jin-qian I-165
Zhang, Hong II-465
Zhang, Hua II-128
Zhang, Jianhong II-395
Zhang, Jianhua II-379
Zhang, Jie II-465
Zhang, Kaibo I-539
Zhang, Kun I-568
Zhang, Liwei I-559
Zhang, Lu I-86
Zhang, Na II-333
Zhang, Wenfang II-379
Zhang, Xing I-439
Zhang, Xuexia I-338
Zhang, Yan-Qing I-522
Zhang, Yi I-310
Zhang, Yongguo II-34
Zhang, Yong-wei I-138
Zhang, Yudong I-514
Zhang, Zili I-364
Zhang, Ziqiu II-57
Zhang, Zulong II-348
Zhao, Bingyan I-416
Zhao, Jing II-457

Zhao, Ling II-183
Zhao, Yan I-578
Zhao, Yong-jian II-157
Zhe, Zhang II-10
Zheng, Dong II-411
Zheng, Mingfa II-34
Zheng, Wang II-108
Zheng, Yihui I-157
Zheng, Yujun II-57
Zhou, Chunguang II-333
Zhou, Huan I-364
Zhou, Jin II-128
Zhou, Lidan I-157
Zhou, Wenyong II-363
Zhou, Xianzhong II-57
Zhou, Zheng I-71
Zhu, Jihong II-502, II-537
Zhu, Jun I-355
Zhu, Peiyi II-494
Zhu, Yuanchun I-382
Zhu, Zexuan I-587
Ziarati, Koorush I-293
Zong, Xinlu I-234
Zuo, Xingquan I-172