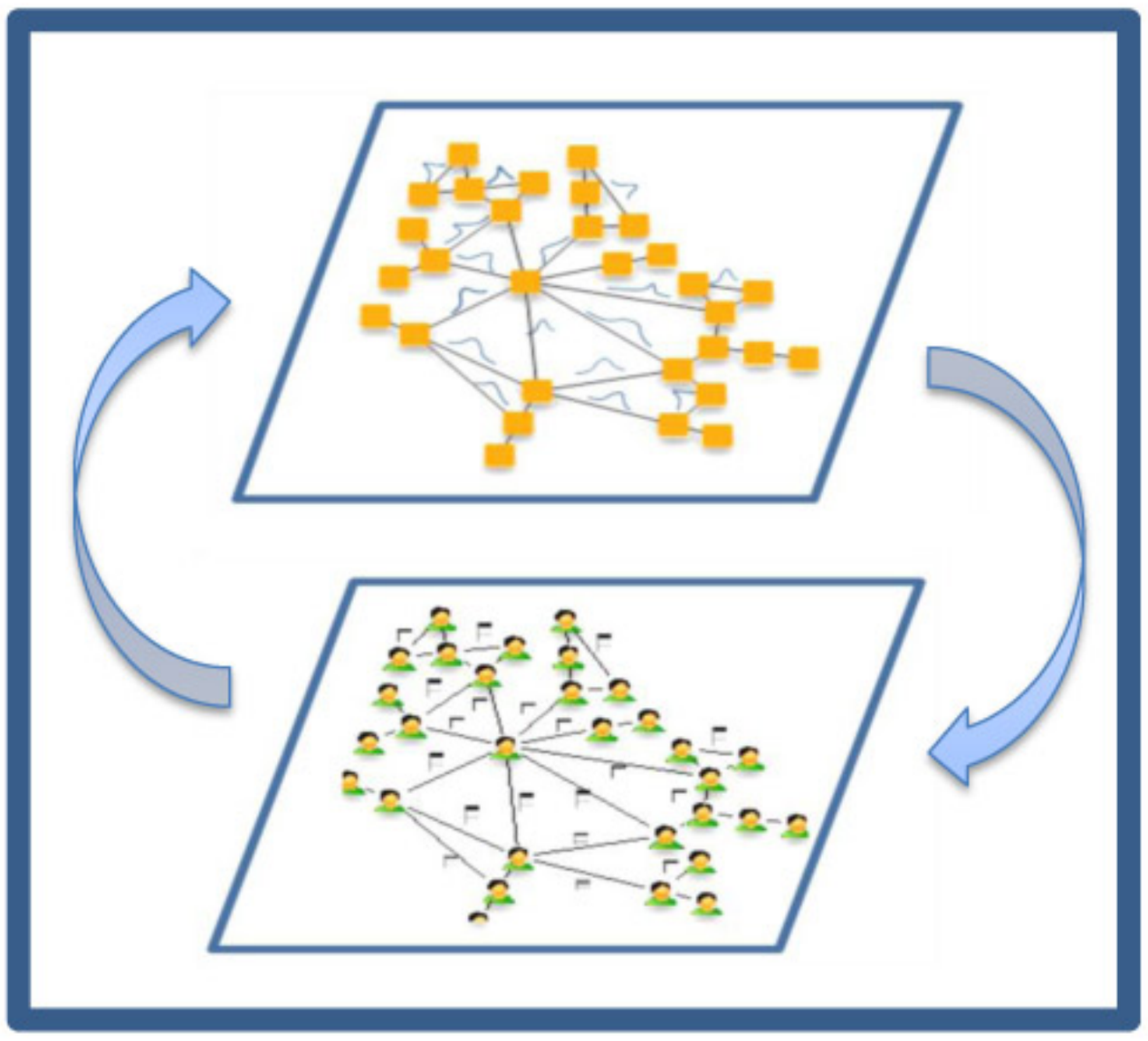
Stochastic Social Networks:
Measures and Algorithms



Alireza Rezvanian and Mohammd Reza Meybodi

Soft computing laboratory, Computer Engineering and Information Technology Department,
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

November 2016

Preface

In most of studies, social networks are modeled and represented as deterministic graphs with a set of nodes as individuals and edges as connection between individuals of networks. However, structural and behavioral parameters of social networks are time-varying parameters due to the uncertain, unpredictable and dynamic nature of users' activities in social networks. Therefore, using deterministic graphs for modeling and analyzing social may not be appropriate. Recently, stochastic graph in which weights associated with edges are random variables is suggested as a better candidate as a graph model for real-world network applications with time-varying nature for social network analysis. By choosing stochastic graph as a graph model of a social network, it is called stochastic social network and also every aspect of network such as concepts, centrality measures, and algorithms are treated stochastically. In this book, we first redefine several networks measures for stochastic social networks and then introduce some intelligent algorithms for computation of network measures for social network analysis under the situation that the weights associated with the edges of the network are random variables with unknown probability distribution functions. Intelligent algorithms can guide the process of sampling the edges of the network in order to provide good estimates for the probability distribution functions of the edges of the network. Finaly, In order to study the performance of the introduced measures and algorithms, experimental studies on different stochastic social networks are also conducted.

Contents

List of Figures III

List of Tables IV

List of Acronyms and Abbreviations V

List of Abbreviation VI

1. Introduction 1

1.1. Organization of the book 3

2. Background 5

2.1. Social networks 5
2.2. User activities in social networks 6
2.3. Stochastic graphs 10
2.4. Network measures for deterministic social networks 11
2.4.1. Degree 11
2.4.2. Strength 11
2.4.3. Closeness 11
2.4.4. Betweenness 12
2.4.5. Clustering coefficient 12
2.5. Learning automata theory 12
2.5.1. Learning automata 13
2.5.2. Fixed-structure learning automata 16
2.5.3. Variable-structure learning automata (VSLA) 16
2.5.4. Learning automata with variable number of actions 16
2.5.5. Network of learning automata 18
2.5.6. Distributed learning automata (DLA) 18
2.5.7. Applications of learning automata 19

3. Network measures in stochastic graphs 21

3.1. Network measures is stochastic graphs 22
3.2. Some algorithms for computing network measures in stochastic graphs 24
3.2.1. Initialization 24
3.2.2. Updating 25
3.2.3. Stopping 25
3.3. Improvements 27
3.3.1. Improvement I 27
3.3.2. Improvement II 27
3.3.3. Improvement III 28
3.3.4. Improvement IV 28
3.3.5. Improvement V 28
3.4. Experiments 30

3.4.1. Evaluation measures 30
3.4.2. Experimental setup 32
3.4.3. Experimental results 33
3.4.4. Discussion 47
3.5. Conclusion and future directions 48
3.5.1. Conclusion 48
3.5.2. Future directions 49
4. Bibliography **50**
A Standard Sampling Method **55**
B Statistical Tests **60**
B.1. Hypothesis testing and *p*-values **60**
B.2. The Wilcoxon matched-pairs signed-ranks test **61**
B.3. The Friedman two-way analysis of variance by ranks **62**
B.4. Post-hoc procedures **63**

List of Figures

Figure 2-1. The interaction of a learning automaton and its environment. 14

Figure 2-2. Pseudo-code of action probability vector update of a learning automaton. 15

Figure 2-3. Pseudo-code of action probability vector update of a variable action-set learning automaton. 18

Figure 2-4. Distributed learning automata. 18

Figure 3-1. Pseudo-code for algorithm 1 for compute network measures in stochastic graphs. 27

Figure 3-2. Pseudo-code for algorithm 6 for computing network measures in stochastic graphs. 29

Figure 3-3. Distributions of betweenness for some randomly selected nodes of the synthetic stochastic BA network. ..34

Figure 3-4. Distributions of clustering coefficient for some randomly selected nodes of the synthetic stochastic BA network. 35

Figure 3-5. Distributions of closeness for some randomly selected nodes of the synthetic stochastic BA network. 36

Figure 3-6. Distributions of strength for some randomly selected nodes of the synthetic stochastic BA network. 37

Figure 3-7. Comparing results of average KS distances, average skew divergence, average normalized L_1 distance and average Pearson's correlation coefficient over betweenness, clustering coefficient, closeness and strength for synthetic stochastic ER graphs. 43

Figure 3-8. Comparing results of average KS distance, average skew divergence, average normalized L_1 distance and average Pearson's correlation coefficient over betweenness, clustering coefficient, closeness and strength for synthetic stochastic BA graphs. 43

Figure 3-9. Comparing results of average KS distance, average skew divergence, average normalized L_1 distance and average Pearson's correlation coefficient over betweenness, clustering coefficient, closeness and strength for synthetic stochastic WS graphs. 44

Figure 3-10. The plot of average cumulative *ESR* versus iteration number for stochastic test graphs. 46

Figure 3-11. Comparison of convergence behaviors of the algorithms with respect to the information entropy for synthetic stochastic graphs. 47

List of Tables

Table 2-1. Summary of studies for user behaviors and their respective distributions. 9

Table 3-1. Average ranking of Friedman's test for randomly selected nodes of the synthetic stochastic BA network. ...37

Table 3-2. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of betweenness of nodes. 38

Table 3-3. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of clustering coefficient of nodes.....38

Table 3-4. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of closeness of nodes. ..38

Table 3-5. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of strength of nodes. 39

Table 3-6. Average results [± standard deviation (std)] for SSM and the algorithms for synthetic stochastic ER graphs with different error rates in terms of ESR. 40

Table 3-7. Average results [± standard deviation (std)] for SSM and the algorithms for synthetic stochastic BA graphs with different error rates in terms of ESR. 40

Table 3-8. Average results [± standard deviation (std)] for SSM and the algorithms for synthetic stochastic WS graphs with different error rates in terms of ESR. 41

Table 3-9. Average ranking of Friedman's test of comparison algorithms on the stochastic test networks. 41

Table 3-10. *p*-values of the comparing algorithms on the stochastic test networks. 41

List of Acronyms and Abbreviations

BA	Barabási-Albert
CC	Clustering Coefficient
CDF	Cumulative Distribution Function
CI	Confidence Intrerval
CI	Confidence Level
DDQC	Degree Distribution Quantification And Comparison
DLA	Distributed Learning Automaton
DT	Dynamic Threshold
ER	Erdös–Rényi
EDLA	Extended Distributed Learning Automata
IM	Influence Maximization
KS	Kolmogorov-Smirnov
KLD	Kullback-Leibler Divergence
LA	Learning Automaton
ND	Normalized Distance
OSN	Online Social Networks
PDF	Probability Density Function
PCC	Pearson's Correlation Coefficient
RE	Relative Error
SD	Skew Divergence
SSM	Standard Sampling Method
VSLA	Variable Structure Learning Automata
WS	Watts–Strogatz

List of Symbols

G	Graph underlying social network
V	Set of nodes
E	Set of edges
$N(u)$	Set of neighboring nodes for node u
m	Number of edges of graph
n	Number of nodes of graph
A_m	learning automaton A_m
α_i^j	j^{th} action of learning automaton A_i
k_i	Degree node v_i
s_i	Strength node v_i
cc_i	Clustering coefficient of node v_i
b_i	Betweenness of node v_i
c_i	Closeness of node v_i

Chapter 1

INTRODUCTION

In recent years, online social networks such as *Facebook* and *Twitter* have provided simple facilities for online users to generate and share a variety of information about users' daily life, activities, events, news and more information about their real worlds. Online social networks similar to many real world networks are usually modeled and represented as deterministic graphs with a set of nodes as individuals and edges as connection between individuals of networks. In social networks analysis, when networks are modeled as weighted or unweight deterministic graphs, one can study, characterize and analyze the statistical characteristics and the dynamical behaviors of the network by some network measures (Costa et al. 2007; Wasserman 1994; Z. Liu et al. 2015). Popular network measures such as degree, betweenness, closeness and clustering coefficient are originally defined for deterministic graphs. These network measures not only used in characterization of networks but also used as a part of some algorithms such as Girvan-Newman community detection algorithm using high betweenness edges (Girvan and Newman 2002), overlapping community detection using nodes' closeness (Badie et al. 2013) and finding the outstanding nodes as a set of seed nodes for maximization of the spread of influence in social network to introduce and promote a new products, services, innovations or technologies (Kempe, Kleinberg, and Tardos 2003), to mention a few.

Online social networks are intrinsically non-deterministic and their structures or behaviors have unpredictable, uncertain and time varying nature and for this reason modeling those using deterministic graph models with fixed weights are too restrictive to solve most of the real network problems and also may lose real information of network due to non-deterministic nature of real world networks. For example in online social networks the activities of users such as friendship behaviors, liking a comment on a post in *Facebook*, visiting user profiles, rating recommendations, asking questionnaires, frequency of notifying events, and frequency of sharing a news, to name a few, vary over time with unknown probabilities (Jin et al. 2013). Analyzing online social networks with deterministic graph models cannot take into consideration the continuum and change of activities of the users occurring over time. Even modeling social networks with weighted graphs in which the edge weights are assumed to be fixed weights considers only a snapshot of a real-world network and is not valid when the tasks/activities on networks vary with time. Moreover, in analyzing online social networks not only understanding the structure and topology of the network is important but the degree of association among the users in network is also important for social network analysis.

In online social networks, studying networks using understanding the central users according to social relationship between users is one of the basic analyses and can be applied for identifying influential users for maximizing the spread of influence through a social network. The central users such as influential spreaders of network may not necessarily be a user with the maximum number of friends at a particular time; rather the behavior of user activities over time is more important than the number of friends of users. For example, many users share their daily life activities such as photos, videos, news or collaborates a scientific paper with their friends or colleagues through online social network services. These shared items and activities of audiences such as their likes, comments, joining to communities and making citations to a scientific paper may change over time and have an impact on the behavior of other users' activities to become interested in favorite items of their friends or collaborators.

Modeling user behavior of online social networks in quantitative manner is important for understanding user interactions and is a challenging task due to the variety of user interactions on the structure of social networks and abundance of data from some online social networks. Several research works regarding modeling user behavior in quantitative manner have been reported in the literatures such as distribution of the number of messages sent by a user in *Facebook* (Golder, Wilkinson, and Huberman 2007), distribution of user's lifetime in *Flickr* and *LinkedIn* (Leskovec et al. 2008), distribution of time gap between link creations in *Facebook* (Nazir, Raza, and Chuah 2008), distribution of members of the communities frequently interact with each other in *Facebook* (Nazir, Raza, and Chuah 2008), distribution of followers in *Twitter* (Kwak et al. 2010), distribution of discussion sizes and user participation levels in Bulletin board system social network (Ding et al. 2010), distribution of number of followers who followed popular contents in *Twitter* and *YouTube* (Falck-Ytter and Øverby 2012), distribution of the number of posts in *Twitter* (Morales, Losada, and Benito 2012), distribution of number of daily active users using some applications in *Facebook* (H. Liu et al. 2013), and distribution of lifetime of tweets in *Twitter* (Bild et al. 2015), to name a few. A summary of the researches conducted on modeling user behavior of online social networks are given in chapter 2.

In this book, according to the aforementioned points, we introduce stochastic social network by suggesting stochastic graph in which weights associated with edges are random variables as a better candidate as a graph model for the real-world network applications with time varying nature. By choosing stochastic graph as a graph model, every aspect of network such as graph problems, network centrality measures, models and algorithms should be treated stochastically. For example, choosing stochastic graph as the graph model of an online social network and defining community structure in terms of clique, and the associations among the individuals within the community as random variables, the concept of stochastic clique may be used to study community structure properties. When an online social network is modeled as stochastic graph, influence spreading using the concept of cover and the influence associations among the users as random variables, the concept of stochastic cover may be more appropriate for investigating spreading properties.

In this book, we first redefine several concepts and definitions such as network centrality measures for stochastic networks which have been already defined for deterministic networks. Then develop learning automata based algorithms for computation of centrality measures under the situation that the weights associated with the edges of the network are random variables with unknown probability distribution functions. Learning automata is utilized in these algorithms to guide the process of sampling the edges of the network in order to provide good estimates for the distribution of the edges and also distribution of the network measures. In the proposed algorithms, the guided sampling process implemented by learning automata aims to take more samples from the promising region of the graph, the regions that reflects higher rate of activities, instead of walking around and taking unnecessary samples from non-promising region of the graph. We show that by using learning automata, the number of samples needed to be taken from the edges of the graph by the guided sampling algorithms is significantly lower than the required number of samples needed to be taken from the edges of the graph when the standard sampling method is used.

In this book, we assume that the structural and behavioral parameters of online social networks are time varying parameters observable by the network. Using the observed parameters, the network computes different measurements (*e.g.*, strength, closeness, betweenness and etc.) in an ongoing fashion. The algorithms introduced in this book can be used by the network as a means for observing the time varying parameters of the network for the purpose of network's measurements computations or any analysis applications. The aim of these algorithms are to collect information as time passes from the network in order to provide good estimates for the network's measurements or analysis applications using fewer numbers of samples than that of standard sampling methods.

1.1. **Organization of the book**

This book consists of seven chapters dedicated towards using stochastic social network for social network analysis. To this end, we redefine a number of concepts and measures in social networks when modeled as stochastic graphs. We also design algorithms for computation of measurements in stochastic graphs when stochastic graphs are used to model social networks.

Chapter 2. This chapter provides the necessary background and gives a brief introduction to social network analysis, stochastic graphs, network measures, user activities in social networks, and the theory of learning automata.

Chapter 3. This chapter argues about the time-varying nature of social network parameters and then redefines the network measures already defined for deterministic network for stochastic network. Also, in this chapter, learning automata-based algorithms for computation of network measures for stochastic graphs will be introduced. The

mathematical studies on the distribution of network measures in stochastic graphs are also given.

Chapter 4. This chapter gives the summary of the book contributions and suggests possible avenues for future directions relevant to the research problems.

Chapter 2

BACKGROUND

In this chapter, to provide the necessary background, we present a brief description of some basic definitions, concepts about social networks, stochastic graphs, network sampling and network measures for deterministic unweighted and weighted networks. We also briefly review the studies performed by the researchers about the stochastic graph problems, network sampling and distributions of user behaviors and activities of users in online social networks. At the end of this section, learning automata and its variants are introduced.

2.1. Social networks

Internet and Web continuously transform the way how information is generated and spread in daily lives of people. In recent years, online social networks (OSN) such as *Facebook* and *Twitter* have been provided simple facilities for online users to generate share and spread a variety of information about users' daily life, activities, events, news and more information about their real worlds through OSN which lead today a large amount of information is being generated and spread in little time. Studying and characterization of the main features of online social networks plays a significant role for analysis of online social networks. For such analysis objectives, online social networks similar to many real world networks are usually modeled and represented as deterministic graphs with a set of nodes as users of social networks and edges as kind of relations between users of social networks. Plenty of studies have been done on the capturing the structural and dynamical characteristics of complex social networks in various applications (Zheng et al. 2015; Wang, Zuo, and Wang 2016; Zhang 2016; Zhao, Li, and Jin 2016) and lots of real networks are found to show the small-world property (Watts and Strogatz 1998), a power-law degree distribution (Watts and Strogatz 1998) or existence of community structures (Fortunato 2010) in networks. The existing research works for studying and analysis of social networks in the literature shows that the most of graph models are usually assumed to be a deterministic graph with fixed weights for its edges or nodes, however online social networks are intrinsically non-deterministic and their structures or behaviors such as online activities of users have unpredictable, uncertain and time varying nature, thus deterministic assumptions for graph model of online social networks with fixed weights are too restrictive to solve most of the real network problems and may lose real information of network due to non-deterministic nature of real world networks. For example in online social networks, the activity behaviors of online users such as following friends, taking a comment on a wall post, liking comments on a given post, posting status posts, visiting user profiles, rating recommendations, asking questionnaires, frequency of notifying events, and frequency of sharing a news, to name a few, vary over time with unknown probabilities (Jin et al. 2013).

Analyzing online social networks with deterministic graph models cannot take into consideration the continuum and variety of activities of the users occurring over time. Even modeling complex social networks with weighted graphs in which the edge or node weights are assumed to be fixed weights considers only a snapshot of a real-world network and it is not valid when the activities and behavior of users on networks vary with time.

In online social networks, studying networks using understanding the central users according to social relationship between users is one of the basic analyses and can be applied for identifying influential users for maximizing the spread of influence through a social network. The central users such as influential spreaders of network may not necessarily be a user with the maximum number of friends at a particular time; rather the behavior of user activities over time is more important than the number of friends of users. For example, many users share their daily life activities such as photos, videos, news or collaborates a scientific paper with their friends or colleagues through online social network services. The shared items and activities of audiences such as their likes, comments, joining to communities and making citations to a scientific paper which may change over time and have an impact on the behavior of other users' activities to become interested in favorite items of their friends or collaborators. In online social networks, the degree of friendship among the users changes directly with the weights of their edges to one another over time, and it is clear that sometimes some edges with high weights are more important than that of some others because they are more accessible and willing to communicate or taking friendship activities. In contrast, there are sometimes the same users are in holidays and have their contributions in their social graph have a less edge weight because they are less accessible and decreased their activities in online social networks.

2.2. **User activities in social networks**

Since, generating and sharing information through social networks are done by users; the online users have become the main features of online social networks. In general, activities of users in most online social networks can be included doing the social relationship with their friends, answering an online questionnaire, joining in online communities, participating in an online discussion, posting and sharing new favorite contents (*e.g.*, photos, music, videos, scientific papers, services, and etc.), visiting a user profile and its contents, taking comment or like on a post. Studying how users behave and interact with their friends in online social networks play a significant role in social network analysis for several reasons (Buccafurri et al. 2015; Shen, Brdiczka, and Ruan 2013; Zhu, Su, and Kong 2015). In some fundamental studies, researchers can generate synthetic graphs similar to real networks in order to further simulations and investigation on a number of phenomena using a computer generated graph models in order to exploit the structures and dynamics of networks and their user behaviors in a systematic manner. In viral marketing, companies can exploit user behavior models and then spread or promote their new products or services to predict the user interactions and process of adoption of an idea or spread them. In some knowledge based researches, understanding the patterns of user participations to generate

new contents and propagate information on the networks via users is very imperative in order to detect active users from spamming users, attract new users and keep some users, predict the trends of topics in user communities, and perform efficient content management in their basic system requirements. Modeling user behavior and human activities of online social networks in quantitative manner is also important for understanding user interactions and is a challenging task due to the variety of user interactions on the structure of online social networks and abundance of data from some online social networks. In recent years, several research works regarding user behavior modeling, characterizing the distribution of user behaviors and user activities and statistical observations from online users have been reported in the literatures, some of which are pointed out in the rest of this section.

Golder et al. (Golder, Wilkinson, and Huberman 2007) analyzed the interactions among *Facebook* users. Their observations discovered strong daily and weekly patterns and found that the communication of the users is clustered mostly based on the universities. They reported that the interactions occurred almost exclusively among friends and only small percentages of friends interact with other friends which results in the distribution of number of messages sent by a user followed a power-law distribution. *Leskovec et al.* (Leskovec et al. 2007) analyzed about 45 thousands blogs and 2.2 million postings, they reported that the popularity of posts drops with a Power-law, instead of exponentially and the size of the cascades distribution follows a perfect Zipfian distribution. *Leskovec et al.* (Leskovec et al. 2008), also focused directly on the microscopic node behavior of four large online social network datasets (*Flickr*, *Delicious*, *Yahoo Answers*, and *LinkedIn*). They observed that user's lifetime follows exponential distribution, time gap between link creations follows a Power-law with exponential cutoff and user activity is uniform over its lifetime or decreases with time exponentially. *Nazir et al.* (Nazir, Raza, and Chuah 2008) presented three applications for *Facebook* to measure the behavior of the users. The authors detected communities, the members of the communities frequently interact with each other and several kinds of human activities were monitored. Based on the measurements, they showed that the popularity of the applications follows a power-law distribution with an exponential decay.

Guo et al. (Guo et al. 2009) analyzed the patterns of user content generation in online social networks. They identified three groups of users. The first group is users with distinct posting behaviors including steadily posting inactively posting, and the second group is users with occasional posting in the network. They found that the overall user lifetime does not follow the exponential distribution and also the distribution of different users' posting activities does not follow Power-law distributions rather it follows the stretched exponential distribution similar to the reference rank distribution of objects in Internet media systems, including viral video social networks such as *YouTube*. *Kwak et al.* (Kwak et al. 2010) crawled the entire *Twitter* site and obtained user profiles, social relations, trending topics and tweets. In their analysis on follower-following topology, they found that the distribution of followers is not power-law; rather the distribution of the users in a Retweet tree follows power-law distribution. *Gyarmati et al.* (Gyarmati and Trinh 2010) analyzed the degree

distribution of nodes for online sessions of users in *Bebo*, *MySpace*, *Netlog*, and *Tagged* based on the crawled public part of users' profile pages, which contained online status information of the users. They revealed that the time spent online by users follows a Weibull distribution that a fraction of users tend to lose interest surprisingly fast soon after subscribing to the service. They claimed that the distribution of session times of users and the number of sessions can be modeled with power-law distributions.

Ding et al. (Ding et al. 2010) studied behaviors of users in the Bulletin board system social network by analyzing the read and reply data and found that the distributions of discussion sizes and user participation levels follows a power-law distribution. *Galuba et al.* (Galuba et al. 2010) analyzed the information propagation in *Twitter* and investigated the propagation of URLs through posts. They found that posting frequencies followed a power-low distribution across users and URLs. Also the authors based on the obtained datasets proposed a model that predicts the propagation of information. *Falck-Ytter et al.* (Falck-Ytter and Øverby 2012) compared the number of followers who followed popular contents in *Twitter* and *YouTube*. Their observations show that Zipf's law was not suitable to describe followers in *Twitter*. In contrast, it was suitable to describe *YouTube* viewers. *Zhong et al.* (Zhong et al. 2012) studied user behavior and interests in different online social networks and observed that they influenced one another. Besides, only small fraction of users actually participated in many activities in online social networks which resulted in a distribution of users with a long-tail property. *Morales et al.* (Morales, Losada, and Benito 2012) analyzed the effects of user behavior on social structure emergence using the information flow on *Twitter* and found that community structure was formed inside the network. In addition, the distribution of the number of posts by users was shown to fit an exponentially truncated Power-law.

Yan et al. (Yan, Wu, and Zheng 2013) explored the behavior of users in posting microblogs systems and realized that the interval time distribution of people posting did not fit a normal distribution, rather it was a power-law distribution. *Liu et al.* (H. Liu et al. 2013) found that the number of daily active users using some applications in *Facebook* follows a power-law distribution. *Feng et al.* (Feng et al. 2013) explored pin distribution (visual bookmarks of images) and board distributions as well as number of comments, likes, repins in *Pinterest* and discovered that those characteristics follows a Power-law. *Vongsingthong et al.* (Vongsingthong et al. 2015) tracked the activities of some users in a *Facebook* through a questionnaire. Their observation on the user activities such as posting photos and videos, visiting popular pages and friend profiles, playing online game, and questioning on products follows a Power-law distribution. *Bild et al.* (Bild et al. 2015), studied empirically aggregate user behavior using an exhaustive observations on *Twitter* with a focus on quantitative descriptions. They found that the lifetime tweet distribution is a type-II discrete Weibull stemming from a Power-law hazard function, that the tweet rate distribution, although asymptotically Power-law, exhibits a lognormal cutoff over finite sample intervals, and that the inter-tweet interval distribution is a Power-law with exponential cutoff.

The summary of studies for type of user behaviors/activities of online social networks and their respective distributions are given in Table 2-1.

Table 2-1. Summary of studies for user behaviors and their respective distributions.

References	Network	Type of user activity or behavior	Type of Distribution
(Kwak et al. 2010)	Twitter	The number of friends	Power-law distribution
(Bild et al. 2015)	Twitter	Tweet rates, Retweet rates	Power-law distribution
(Guo et al. 2009)	Answers	Overall user lifetime, different users' posting activities	Stretched exponential distribution
(Golder, Wilkinson, and Huberman 2007)	Facebook	Number of messages sent by a user	Power-law distribution
(Leskovec et al. 2008)	Delicious, Flickr, Answers, LinkedIn	User's lifetime	Exponential distribution
(Nazir, Raza, and Chuah 2008)	Facebook	Popularity of the social applications among users	Power-law distribution with an exponential decay
(Nazir, Raza, and Chuah 2008)	Facebook	Number of unique users that visit the application least once during a given day	Power-law distribution with an exponential cutoff
(Cha et al. 2007)	YouTube, Daum	The popularity distribution of user generated video	Power-law distribution with an exponential cutoff
(Gyarmati and Trinh 2010)	Bebo, MySpace, Netlog, Tagged	Time spent online by users	Weibull distribution
(Gyarmati and Trinh 2010)	Bebo, MySpace, Netlog, Tagged	Session times of users The number of sessions	Power-law distribution
(Ding et al. 2010)	TianYa BBS	Discussion sizes and user participation levels	Power-law distribution
(Falck-Ytter and Øverby 2012)	YouTube	YouTube viewers	Zipf's law distribution
(Morales, Losada, and Benito 2012)	Twitter	The number of posts by users	Exponentially truncated Power-law distribution
(Yan, Wu, and Zheng 2013)	Sina	Interval time distribution of people posting	Power-law distribution
(H. Liu et al. 2013)	Facebook	The number of daily active users using applications	Power-law distribution
(Feng et al. 2013)	Pinterest	Pin distribution, visual bookmarks of images, board distributions, number of comments, likes and repins	Power-law distribution
(Vongsingthong et al. 2015)	Facebook	Posting photos and videos, visiting popular pages and friend profiles, Playing online game, and questioning on products	Power-law distribution
(Galuba et al. 2010)	Twitter	Posting frequencies across users and URLs	Power-law distribution
(Bild et al. 2015)	Twitter	Lifetime tweet	Type-II discrete Weibull

(Bild et al. 2015)	Twitter	Inter-tweet interval	distribution Power-law distribution with exponential cutoff

2.3. **Stochastic graphs**

A weighted undirected graph can be described by a triple $G=\langle V, E, W\rangle$ where $V=\{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $E=\{e_{ij}\}\in V\times V$ is the edge-set in which e_{ij} indicates a connection between node v_i and node v_j, and W is a matrix in which w_{ij} is a weight associated with the edge between nodes v_i and v_j if such an edge exists. Graph G is said to be stochastic if weight w_{ij} associated with edge e_{ij} is a random variable with the probability density function (PDF) q_{ij}.

The set of nodes V can represent users in social networks, authors in citation networks, friends in friendship networks or airline terminal in transportation networks. An edge in edge-set E can represent a type of a connection, relationship, task, activity or flow which can be generated, produced, transferred or terminated through the edges of graph. The weights may be defined according to the tasks/activities on an edge between two nodes such as strength activities between two users in online social networks that it is not always deterministic, the number of collaborations (co-authoring a paper) between scientist authors that may change over time, the amount of trust between friends that varies with time or the links in a communication network that may be affected by collisions, congestions, interferences or some other factors. The weights of connections in real world networks activities for each mentioned process may vary over time. Hence, it seems better to represent these networks as stochastic graphs where the weights associated with the nodes or edges are random variables rather than deterministic graphs.

Torkestani et al. (J. A. Torkestani and Meybodi 2012a) designed a learning automata-based heuristic algorithm to solve the minimum spanning tree problem in stochastic graphs in such a way that the algorithm determines the edges that must be sampled at each stage. Furthermore, he defined a variant version of spanning tree problems as the stochastic min-degree constrained minimum spanning tree problem and then proposed a decentralized learning automata-based heuristic to solve it (Javad Akbari Torkestani 2013).

Torkestani et al. presented several learning automata-based algorithms to solve the minimum weakly connected dominating set problem in a stochastic graph (J. Torkestani and Meybodi 2010). Also they introduced the stochastic version of the minimum connected dominating set problem in vertex weighted graphs, then, they proposed several learning automata-based algorithms for solving the problem. They claimed that their algorithm guaranteed neither the domination of all the graph nodes nor the connection of the dominators (J. A. Torkestani and Meybodi 2012b). *Moradabadi* et al. modeled the time series link prediction problem in online social networks as a stochastic graph problem by considering different similarity metrics and link occurrence over time, concurrently. Their solution is established based on a team of continuous action set learning automata for solving a stochastic graph problem (B. Moradabadi and Meybodi 2016).

2.4. **Network measures for deterministic social networks**

Network measures and computing them play a significant role in social network analysis (Borgatti 2005). Popular network measures such as degree, betweenness, closeness and clustering coefficient not only used for evaluating the node importance in actual complex network studies but also used as a part of some algorithms such as Girvan-Newman community detection algorithm using betweenness (Girvan and Newman 2002), overlapping community detection using node closeness (Badie et al. 2013). In this section some of well-known network measures for deterministic networks are introduced.

2.4.1. **Degree**

Degree as a basic network measure has been widely used in many studies and degree of node v_i defined in binary network (also called unweighted network) as follows

$$k_i = \sum_{j \neq i} a_{ij} \tag{2-1}$$

where j is the index of all other nodes of graph and a_{ij} is 1 if node v_i is adjacent to node v_j, and 0 otherwise. In other words, degree of node v_i is the number of nodes that directly connected to node v_i. Degree centrality is useful in the context of finding the single human which gets affected by the diffusion of any information in the network. It follows from the fact that the human with high degree centrality has the chance of getting affected from many numbers of sources (Freeman 1979).

2.4.2. **Strength**

Node strength of node v_i is defined as the sum of adjacent edge weights for weighted network as follows

$$s_i = \sum_{j \neq i} w_{ij} \tag{2-2}$$

where w_{ij} is greater than 0 if node v_i is adjacent to node v_j and its value indicates the weight of edge between node v_i and node v_j. Similar to degree, a human with high strength centrality is known as popular human with high strength links to other humans; however a human with high strength may not consist of necessarily the maximum number of friends. Strength centrality is useful in the context of finding a human which gets affected by the amount of spreading of any information in the network (Freeman 1979).

2.4.3. **Closeness**

Closeness of a node is the inverse sum of shortest paths to all other nodes from that node and defined for binary network with n nodes as follows

$$c_i = \frac{1}{\sum_{j \neq i} d_{ij}} \tag{2-3}$$

where d_{ij} is the length of shortest path between node v_i and node v_j. Closeness can be regarded as a measure of how long it will take to spread information from that node to all

other nodes sequentially. Since, the spread of information can be modeled by the use of shortest paths, in applications such as spread of information, a human with high closeness centrality can be considered as the central point because that human can spread the information faster than other human with lower closeness centrality (Freeman 1979).

2.4.4. **Betweenness**

Betweenness of a node is the number of shortest paths from all nodes to all other nodes of graph that pass through that node. Betweenness of node v_i is defined for binary network as follows

$$b_i = \frac{g_{st}(i)}{g_{st}} \tag{2-4}$$

where g_{st} is the number of shortest paths between all pair of nodes of source node v_s and destination node v_t in a binary network and $g_{st}(i)$ is the number of those paths that pass through node v_i. Since, a node with high betweenness centrality will typically be the one which acts as a bridge between many pairs of nodes, the betweenness introduced as a measure for quantifying the control of a human on the communication between other humans in a social network. In this conception, humans that have a high probability to occur on a randomly chosen shortest path between two randomly chosen humans have a high betweenness. Betweenness can be regarded as a measure of how to control Information flows over communication links. Also, Information flows could be dominated by humans with high betweenness centrality in a network (Freeman 1979; Newman 2005).

2.4.5. **Clustering coefficient**

The clustering coefficient for node v_i is defined as a proportion of the number of all edges among its neighbors over the possible number of edges between them. The local clustering coefficient for a node v_i in binary network can be defined as follows

$$CC(i) = \frac{1}{k_i(k_i - 1)} \sum_{v_j, v_h \in N_i} a_{ij}.a_{ih}.a_{jh} \tag{2-5}$$

where k_i is the degree of node v_i, n is the total number of nodes of graph, N_i is the set of nodes v_j adjacent to node v_i and a_{ij} is 1 if node v_i is adjacent to node v_j, and 0 otherwise. The clustering coefficient of a node is a measure of the connectivity among the neighborhood of the node (transitivity of a node). A human with high clustering coefficient indicates the high tendency of that human to form cluster with other humans. In other words, the clustering coefficient measures transitivity of a network. And also, most of the friends of a human with high clustering coefficient can collaborate with one another even if the focal human is removed from the network.

2.5. **Learning automata theory**

Reinforcement learning or learning with a critic is defined by characterizing a learning problem, instead of characterizing the learning methods (Sutton and Barto 1998). Any

method that is well suited to solve that problem is considered as a reinforcement learning method. The reinforcement learning problem is the problem of learning from interactions in order to achieve a certain goal. Considering this specification for the reinforcement learning problem, there must be an agent capable of learning, called the learner or the decision-maker. The learner must interact with a so-called environment or teacher, comprising everything outside of the learner, which provides evaluative responses to the actions performed by the learner. The learner and the environment interact continually; the learner selects actions and the environment responds to the selected actions, presenting new situations to the learner. In short, reinforcement learning is a framework of the learning problems in which the environment does not indicate the correct action, but provides only a scalar evaluative response to the selection of an action by the learner.

2.5.1. **Learning automata**

Learning automaton (LA) as one of computational intelligence techniques have been found very useful tool to solve many complex and real world problems in networks where a large amount of uncertainty or lacking the information about the environment exists (A. Mousavian, Rezvanian, and Meybodi 2013; J. A. Torkestani 2012; Mahdaviani et al. 2015; Aylin Mousavian, Rezvanian, and Meybodi 2014; Khomami, Rezvanian, and Meybodi 2016; Mirsaleh and Meybodi 2016a; Zhao et al. 2015). A learning automaton is a stochastic model operating in the framework of the reinforcement learning (Kumpati S. Narendra and Thathachar 1989; M. A. L. Thathachar and Sastry 2004). This model can be classified under the reinforcement learning schemes in the category of the temporal-difference (*TD*) learning methods. *TD* learning is a combination of the Monte Carlo ideas and the dynamic programming ideas. Like Monte Carlo methods, *TD* methods can learn directly from raw experience without a model of the environment's dynamics. Like the dynamic programming, *TD* methods update estimates based in part on the other learned estimates, without waiting for a final outcome (Sutton and Barto 1998). Sarsa (Rummery and Niranjan 1994), Q-learning (Watkins 1989), Actor-Critic methods (Barto, Sutton, and Anderson 1983) and R-learning (Schwartz 1993) are other samples of *TD* methods. Learning automata differ from other *TD* methods in the following two ways; the representation of the internal states and the updating method of the internal states.

The automata approach to learning can be considered as the determination of an optimal action from a set of actions. A learning automaton can be regarded as an abstract object that has a finite number of actions. It selects an action from its finite set of actions and applies it to an environment. The environment evaluates the applied action and sends a reinforcement signal to the learning automaton (Figure 2-1). The reinforcement signal provided by the environment is used to update the internal state of the learning automaton. By continuing this process, the learning automaton gradually learns to select the optimal action, which leads to favorable responses from the environment.

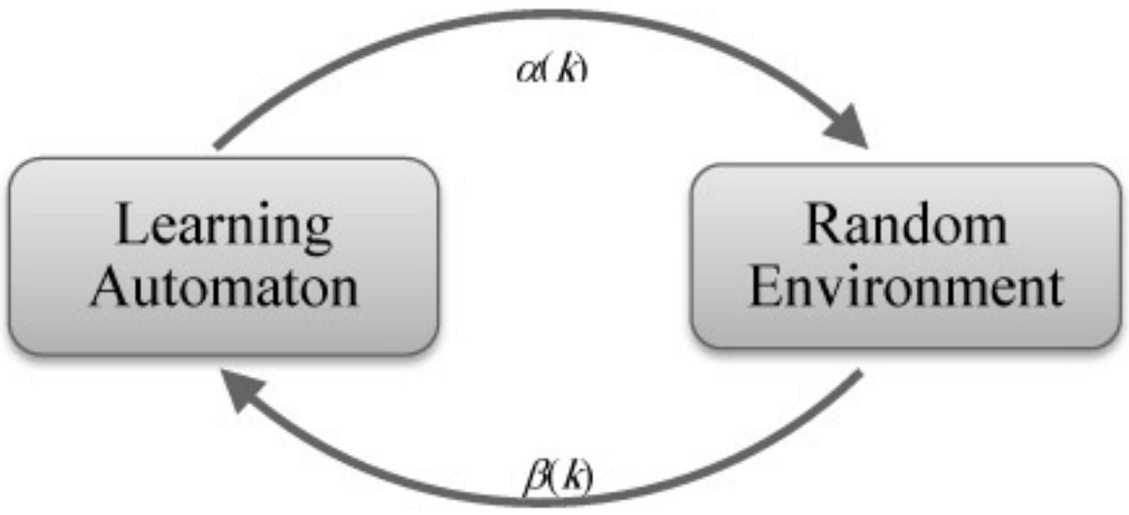


Figure 2-1. The interaction of a learning automaton and its environment.

A learning automaton is a quintuple $\langle \alpha, \Phi, \beta, F, G \rangle$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of actions that it must choose from, $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_s)$ is the set of states, $\beta = \{\beta_1, \beta_2, \ldots, \beta_q\}$ is the set of inputs, G: $\Phi \rightarrow \alpha$ is the output map and determines the action taken by the learning automaton if it is in the state Φ_j, and F: $\Phi \times \beta \rightarrow \Phi$ is the transition map that defines the transition of the state of the learning automaton upon receiving an input from the environment.

The selected action at the time instant k, denoted by $\alpha(k)$, serves as the input to the environment which in turn emits a stochastic response, $\beta(k)$, at the time instant k, which is considered as the response of the environment to the learning automaton. Based upon the nature of β, environments could be classified in three classes: P-, Q-, and S-models. The output of a P-model environment has two elements of success or failure. Usually in P-model environments, a failure (or unfavorable response) is denoted by 1 while a success (or a favorable response) is denoted by 0. In Q-model environments, β can take a finite number of values in the interval [0,1] while in S-model environments, β lies in the interval [0, 1]. Based on the response $\beta(k)$, the state of the learning automaton $\Phi(k)$ is updated and a new action is chosen at the time instant $(k+1)$. Learning automata can be classified into two main classes: fixed and variable structure learning automata (VSLA) (Kumpati S. Narendra and Thathachar 1989) which is used in this book.

A simple pseudo-code for updating probabilities of a learning automaton in a stationary environment with $\beta \in \{0,1\}$ and r actions is presented in Figure 2-2.

Algorithm 2-1. Learning automaton

Input: Action-set α

Output: Action probability vector p

Assumptions

Initialize r-dimensional action-set: $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ with r actions

Initialize r-dimensional action probability vector $p(k) = \{p_1, p_2, \ldots, p_r\} = \left\{\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r}\right\}$ at instant k

```
Let i denotes the selected action by the automaton
Let j denotes the current checking action
Begin
  While (automaton converge to an action)
    The learning automaton selects an action based on the action probability vector p.
    The environment evaluates the selected action and gives the reinforcement signal β∈{0, 1} to the
      learning automaton.
    Update the probability vector p(k)
    For each action j∈{1, ... , r} Do
      If (β=0) // favorable response
          The selected action by learning automaton is rewarded according to equation (2-6)
      Else If (β=1) // unfavorable response
          The selected action by learning automaton is penalized according to equation (2-7)
      End If
    End For
  End While
End Algorithm
```

Figure 2-2. Pseudo-code of action probability vector update of a learning automaton.

Learning automata have several good features, which make them suitable for using in many applications. Main features of learning automata are stated below

1. Learning automata can be used without any priori information about the underlying application.
2. Learning automata are very useful for applications with large amount of uncertainty.
3. Unlike traditional hill-climbing algorithms, hill-climbing in learning automata is done in expected sense in a probability space.
4. Learning automata require a very little and simple feedback from their environment.
5. Learning automata are very useful in multi-agent and distributed systems with limited intercommunication and incomplete information.
6. Learning automata are very simple in structure and can be implemented easily in software or hardware.
7. The action set of learning automata can be a set of symbolic or numeric values.
8. Optimization algorithms based on learning automata don't need the objective function to be an analytical function of adjustable parameters.
9. Learning automata can be analyzed by powerful mathematical methodologies. It has been shown that learning automata are optimal in single, hierarchical, and distributed structures.
10. Learning automata require a few mathematical operations at each iteration so it can be used in real-time applications.
11. Learning automata have flexibility and analytical tractability needed for most applications.

2.5.2. **Fixed-structure learning automata**

The learning automaton is called fixed-structure if the followings are fixed: 1. the probability of the transition from one state to another state and 2. the action probability of any action in any state. Examples of the fixed-structure *LA* are L_2N_2, G_2N_2, Krylov, and Krinsky (Kumpati S. Narendra and Thathachar 1989) which has been used in many applications such as adaptation of the back-propagation algorithm parameters.

2.5.3. **Variable-structure learning automata (VSLA)**

A VSLA is represented by a 6-tuple $\langle \beta, \phi, \alpha, P, G, T \rangle$, where β denotes the set of inputs, ϕ is a set of internal states, α is the action sets as output, P is the state probability vector governing the choice of state as each instant k, G is the output mapping, and T is the learning algorithm (also known as learning scheme). The learning algorithm T refers to a recurrence relation which is used to update the action probability vector. Let $\alpha_i(k) \in \alpha$ be the action that is chosen by a learning automaton and $p(k)$ is the probability vector defined over the set of action at instant k. At each instant k, the action probability vector $p(k)$ is updated by the linear learning algorithm given in equation (2-6) if the chosen action $\alpha_i(k)$ is rewarded by the random environment (β=0) and it is updated according to equation (2-7) if the chosen action is penalized (β=1).

$$p_j(k+1) = \begin{cases} p_j(k) + a\left[1 - p_j(k)\right] & j = i \\ (1-a)p_j(k) & \forall j \neq i \end{cases} \tag{2-6}$$

$$p_j(k+1) = \begin{cases} (1-b)p_j(k) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(k) & \forall j \neq i \end{cases} \tag{2-7}$$

where a denotes reward parameter which determines the amount of increases of the action probability values, b is the penalty parameter determining the amount of decrease of the action probabilities values and r is the number of actions that learning automaton can take. If a=b, the learning algorithm is called linear reward-penalty (L_{RP}) algorithm; if b=εa where $0<\varepsilon<1$, then the learning algorithm is called linear reward-ε-penalty ($L_{R\varepsilon P}$) algorithm; and finally if b=0, the learning algorithm is called linear reward-Inaction (L_{RI}) algorithm which is perhaps the earliest scheme considered in mathematical psychology.

2.5.4. **Learning automata with variable number of actions**

In variable action-set learning automata (also called a learning automaton with variable number of actions) the number of available actions that can be taken by the learning automaton at every instant can be changed with time (M. Thathachar 1987). In variable action set learning automata, at each instant k, a learning automaton selects an action based on the scaled action probability vector $\hat{p}(k)$ defined as

$$\hat{p}_i(k) = \frac{p_i(k)}{K(k)} \tag{2-8}$$

where $K(k)$ is the sum of the probabilities of the available actions at instant k, and $p_i(k) = Prob[\alpha(k) = \alpha_i]$. And then the selected action performed on the environment. After receiving the reinforcement signal, the LA will similarly update probability vector of available actions via learning algorithm given in equations (2-6) and (2-7). Finally, the probability vector of available actions $p(k+1)$ is rescaled as follows

$$p_i(k+1) = \hat{p}_i(k+1).K(k) \tag{2-9}$$

The pseudo-code of probability update of a variable action set learning automaton is shown in Figure 2-3.

Algorithm 2-2. Variable action-set learning automata

Input: Action-set α
Output: Action probability vector p
Assumptions
Initialize r-dimensional action-set: $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ with r actions
Initialize r-dimensional action probability vector $p(k) = \{p_1, p_2, \ldots, p_r\} = \left\{\frac{1}{r}, \frac{1}{r}, \ldots, \frac{1}{r}\right\}$ at instant k
Let i denotes the selected action by the automaton
Let j denotes the current checking action
Begin
While (automaton converge to an action)
Calculate available actions of learning automaton
Calculate the sum of the probability of available actions
For each action $j \in \{1, \ldots, r\}$ **Do**
If (α_j is available action)
Scale action probability vector $\hat{p}_i(k)$ according to equation (2-8)
End if
End for
Generate map function between available actions and all actions of learning automaton
The learning automaton selects an action according to the probability vector of available actions $\hat{p}(k)$
The environment evaluates the selected action and gives the reinforcement signal $\beta \in \{0, 1\}$ to the learning automaton.
For each available action $j \in \{1, \ldots, m\}$ **Do**
If (β=0) // favorable response
The selected action by learning automaton is rewarded according to equation (2-6)
Else If (β=1) // unfavorable response
The selected action by learning automaton is rewarded according to equation (2-7)
End if
End for
For each action $j \in [1, \ldots, r]$ do
If (α_j is available action)
Rescale the probability vector of selected available action by equation (2-9)
End
End for
End While
End Algorithm

Figure 2-3. Pseudo-code of action probability vector update of a variable action-set learning automaton.

2.5.5. **Network of learning automata**

A network of *LA*s (Williams 1988) is a collection of *LA*s connected together as a hierarchical feed-forward layered structure. In this structure, the outgoing link of the *LA*s in the preceding layers is the input of the *LA*s of the succeeding layers. In this model, the *LA*s (and consequently the layers) are classified into three separate groups. The first group includes the *LA*s located at the first level of the network, called the input *LA*s (input layer). The second group is composed of the *LA*s located at the last level of the network, called the output *LA*s (output layer). The third group includes the *LA*s located between the first and the last layers, called the hidden *LA*s (hidden layer). In a network of *LA*s, the input *LA*s receive the context vectors as external inputs from the environment, and the output *LA*s apply the output of the network to the environment. This model operates as follows. The context vector is applied to the input *LA*s. Each input *LA* selects one of its possible actions on the basis of its action probability vector and the input signals it receives from the environment. The chosen action activates the *LA*s of the next level, which are connected to this *LA*. Each activated *LA* selects one of its actions as stated before. The actions selected by the output *LA*s are applied to the random environment. The environment evaluates the output action in comparison with the desired output and generates the reinforcement signal. This reinforcement signal is then used by all *LA*s for updating their states.

2.5.6. **Distributed learning automata (DLA)**

A Distributed learning automata (DLA) (Beigy and Meybodi 2006) shown in Figure 2-4 is a network of interconnected learning automata which collectively cooperate to solve a particular problem. The number of actions for a particular LA in DLA is equal to the number of LA's that are connected to this LA. Selection of an action by a LA in DLA activates another LA which corresponds to this action. Formally, a DLA can be defined by a quadruple $\langle A, E, T, A_0 \rangle$, where $A=\{A_1, A_2, \ldots, A_n\}$ is the set of learning automata, $E \subset A \times A$ is the set of edges where edge (v_i, v_j) corresponds to action α_i^j of automaton A_i, T is the set of learning algorithms with which learning automata update their action probability vectors, and A_0 is the root automaton of DLA at which activation of DLA starts.

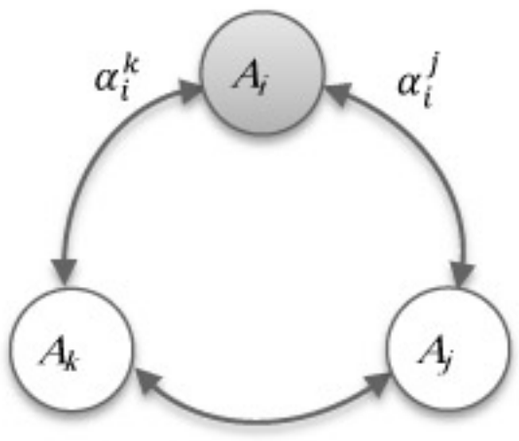


Figure 2-4. Distributed learning automata.

The operation of a DLA can be described as follows: At first, the root automaton A_0 randomly chooses one of its outgoing edges (actions) according to its action probabilities and activates the learning automaton at the other end of the selected edge. The activated automaton also randomly selects an action which results in activation of another automaton. The process of choosing actions and activating automata is continued until a leaf automaton (an automaton which interacts with the environment) is reached. The chosen actions, along the path induced by the activated automata are applied to the random environment. The environment evaluates the applied actions and emits a reinforcement signal to DLA. The activated learning automata along the chosen path update their action probability vectors on the basis of the reinforcement signal according to the learning algorithms. The paths from the unique root automaton to one of the leaf automata are selected until the probability with which one of the chosen paths is close enough to unity. Each DLA has exactly one root automaton which is always activated, and at least one leaf automaton which is activated probabilistically. For example in Figure 2-4, every automaton has two actions. If automaton A_0 selects α_0^2 from its action set, then it will activate automaton A_2. Afterward, automaton A_2 chooses one of its possible actions and so on.

2.5.7. **Applications of learning automata**

In the recent years, learning automata have been used as optimization tools in complex and dynamic environments where a large amount of uncertainty or lacking the information about the environment exists. In the recent years, learning automata have been successfully applied to a wide variety of applications such as optimization (M Hasanzadeh, Meybodi, and Ebadzadeh 2013; Behnaz Moradabadi and Beigy 2014; Mahdaviani et al. 2015; Behnaz Moradabadi, Ebadzadeh, and Meybodi 2016; Vafashoar and Meybodi 2016), (Behnaz Moradabadi and Beigy 2014), image processing (Mofrad et al. 2015; Damerchilu, Norouzzadeh, and Meybodi 2016), graph problems (A. Mousavian, Rezvanian, and Meybodi 2013; Aylin Mousavian, Rezvanian, and Meybodi 2014; Mirsaleh and Meybodi 2016b), community detection (Amiri et al. 2013; Khomami, Rezvanian, and Meybodi 2016; Mirsaleh and Meybodi 2016a), link prediction (B. Moradabadi and Meybodi 2016), grid computing (M Hasanzadeh and Meybodi 2014; Mohammad Hasanzadeh and Meybodi 2015), wireless sensor networks (Safavi, Meybodi, and Esnaashari 2014; Nicopolitidis 2015), WiMAX networks (Misra, Chatterjee, and Guizani 2015), network security (Krishna et al. 2014), wireless mesh networks (N. Kumar and Lee 2015), mobile video surveillance (Narendra Kumar, Lee, and Rodrigues 2015), vehicular environment (N. Kumar et al. 2014; N. Kumar, Misra, and Obaidat 2015), Peer-to-Peer networks (Saghiri and Meybodi 2016), and cloud computing (Morshedlou and Meybodi 2014).

Chapter 3

NETWORK MEASURES IN STOCHASTIC GRAPHS

In recent years, online social networks such as *Facebook* and *Twitter* have provided simple facilities for online users to generate and share a variety of information about users' daily life, activities, events, news and more information about their real worlds which results in the online users have become the main features of online social networks and studying how users behave and interact with their friends in online social networks play a significant role for analysis of online social networks. Online social networks similar to many real world networks are usually modeled and represented as deterministic graphs with a set of nodes as users and edges as connection between users of networks. In social networks analysis, when networks are modeled as weighted or unweight deterministic graphs, one can study, characterize and analyze the statistical characteristics and the dynamical behaviors of the network and its user behaviors by some network measures (Costa et al. 2007). Popular network measures such as degree, betweenness, closeness and clustering coefficient are originally defined for deterministic graphs. These network measures not only used in characterization of networks but also used as a part of some algorithms such as Girvan-Newman community detection algorithm using high betweenness edges (Girvan and Newman 2002), overlapping community detection using nodes' closeness (Badie et al. 2013) and finding the outstanding nodes as a set of seed nodes for maximization of the spread of influence in social network aims to introduce and promote a new products, services, innovations or technologies by ranking important humans (Li et al. 2014).

Online social networks are intrinsically non-deterministic and their structures or behaviors such as online activities of users have unpredictable, uncertain and time varying nature and for this reason modeling those using deterministic graph models with fixed weights are too restrictive to solve most of the real network problems. For example in online social networks the online activities of users such as friendship behaviors, liking a comment on a given post in *Facebook* and frequency of taking a comment on a wall post vary over time with unknown probabilities (Jin et al. 2013). Analyzing online social networks with deterministic graph models cannot take into consideration the continuum of activities of the users occurring over time. Even modeling social networks with weighted graphs in which the edge weights are assumed to be fixed weights considers only a snapshot of a real-world network and it is not valid when the activities and behavior of users on networks vary with time. Moreover, in analyzing online social networks not only understanding the structure and topology of the network is important but the degree of association among the users in network is also important for analysis of user behaviors in online social networks.

According to the aforementioned points, it seems that stochastic graphs in which weights associated with the edges are random variables is a better candidate as a graph model for real-world network applications with time varying nature. By choosing a stochastic graph as a graph model, every feature, measures and concept of the graph such as path (Beigy and Meybodi 2006), cover, clique, and spanning tree (J. A. Torkestani and Meybodi 2012a) should be treated as stochastic features. For example, choosing stochastic graph as the graph model of an online social network and defining community structure in terms of clique, and the associations among the humans within the community as random variables, the concept of stochastic clique may be used to study community structure properties.

In this chapter, we first redefine some network measures for stochastic graphs and then design six algorithms for computing them under the situation that the probability distribution functions of the weights associated with the edges are unknown. The introduced algorithms for computing network measures by taking samples from the edges of the stochastic graph try to estimate the distribution of the network measures. The process of sampling from the edges of the graph is guided by the aid of learning automata in such a way that the number of samples needed to be taken from the edges of the stochastic graph for estimating the network measures to be reduced as much as possible. In the introduced algorithms, the guided sampling process implemented by learning automata aims to take more samples from the promising region of the graph, the regions that reflects higher rate of changes (*i.e.*, higher rate of user activities), instead of walking around and taking unnecessary samples from non-promising region of the graph.

In order to study the performance of the introduced algorithms for computing network measures in stochastic graphs, several experimental studies on different synthetic stochastic graphs are conducted. Experimental results show that in order to obtain good estimates for the network measures the number of samples needed to be taken from the edges of the graph by the guided sampling algorithms is significantly lower than the required number of samples needed to be taken from the edges when standard sampling method is used aims to analysis of human activities and user behaviors in online social networks.

The rest of this chapter is organized as follows. In Section 3.1, the network measures for stochastic graphs are described. Section **Error! Reference source not found.** is dedicated to mathematical analysis of the network measures in stochastic graphs. The introduced algorithms for computing the network measures in stochastic graphs are described in section 3.2 and section 3.3 gives some improvements on the first algorithm for computing the network measures in stochastic graphs. Section 3.4 presents the simulation results. Finally, section 3.5 concludes the chapter.

3.1. Network measures is stochastic graphs

In this section, we first define some concepts in stochastic graphs such as path and spanning tree and then define network measures: strength, closeness, betweenness and clustering coefficient for analysis of stochastic graphs.

Definition 3.1 (*Stochastic graph*). A stochastic graph G can be described by a triple $G=\langle V, E, W\rangle$, where $V=\{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $E=\{e_{ij}\}\subseteq V\times V$ is the edge-set, and W is a matrix in which w_{ij} is a random variable associated with edge e_{ij} between node v_i and node v_j if such that edge exists.

Definition 3.2 (*Stochastic path*). In a stochastic graph, a *path* π_i with weight of w_{π_i} and length of n_i from source node v_s to destination node v_d, can be defined as an ordering $\{\pi_1^i, \pi_2^i, \ldots, \pi_{n_i}^i\}\subset V$ of nodes in such a way that $\pi_1^i = v_s$ and $\pi_{n_i}^i = v_s$ are source and destination nodes, respectively and edge $e(\pi_{j-1}^i, \pi_j^i)\in E$ for $1 \le j \le n_i$, where π_j^i is the j^{th} node in path π_i.

Definition 3.3 (*Stochastic shortest path*). Let in a stochastic graph, there are r distinct paths $\Pi=\{\pi_1, \pi_2, \ldots \pi_r\}$ between source node v_s and destination node v_d, *path* π^* between source node v_s and destination node v_d is defined as a stochastic shortest path if and only if $w_{\pi^*} = \min_{\pi_i\in\Pi}\{w_{\pi_i}\}$ where $w_{\pi_i} = \sum_{e_{jk}\in\pi_i} w_{jk}$ is the expected weight of path $\pi_i\in\Pi$ and called *stochastic shortest path*. That is a stochastic shortest path is a stochastic path with the minimum expected weight.

Definition 3.4 (*Minimum spanning tree*). Let in a stochastic graph G, $\Theta=\{\tau_1, \ldots,\tau_n\}$ be the set of all possible spanning trees of graph G and $w_{\tau_i} = \sum_{e_{jk}\in\tau_i} w_{jk}$ be the weight of spanning tree τ_i, where w_{jk} is the edge weight of edge $e_{jk}\in E$. Therefore, spanning tree τ^* is the minimum spanning tree of graph G if and only if $w_{\tau^*} = \min_{\tau_i\in\Theta}\{w_{\tau_i}\}$.

Definition 3.5 (*Stochastic minimum spanning tree*). Let in a stochastic graph, Θ is the set of spanning trees of stochastic graph, stochastic spanning tree τ^* is called stochastic minimum spanning tree if and only if $w_{\tau^*} = \min_{\tau_i\in\Theta}\{w_{\tau_i}\}$ where w_{τ_i} is the expected weight of spanning tree $\tau_i\in\Theta$. That is a stochastic minimum spanning tree is a stochastic tree with the minimum expected weight.

Definition 3.6 (*Stochastic strength*). The strength of a node v_i in a stochastic graph is a random variable defined by following equation

$$SS(v_i) = \sum_{j\neq i} \overline{w}_{ij} \tag{3-1}$$

where $\overline{w}_{ij}$ is a random variable associated with the edge weight between node v_i and node v_j.

Definition 3.7 (*Stochastic closeness*). The closeness of node v_i in a stochastic graph is a random variable defined by equation (3-2)

$$SC(v_i) = \frac{1}{\sum_j D_{ij}} \tag{3-2}$$

where ED_{ij} is a random variable associated with the weight of shortest path between node v_i and node v_j with minimum expected weight.

Definition 3.8 (*Stochastic betweenness*). The betweenness of node v_i in a stochastic graph is a random variable SB_i defined by equation (3-3)

$$SB(v_i) = \frac{G_{st}(v_i)}{G_{st}} \tag{3-3}$$

where $G_{st}(v_i)$ is the distribution of the sum of weight of shortest paths between node v_s and node v_t that passes through node v_i and G_{st} is the distribution of the sum of weights of all shortest paths between node v_s and node v_t.

Definition 3.9 (*Stochastic clustering coefficient*). The clustering coefficient of a node v_i in a stochastic graph is a random variable which can be defined as follows

$$SCC(v_i) = \frac{1}{(D(v_i)-1).SS(v_i)} \sum_{j \neq i} \sum_{k \neq i,j} \frac{(w_{ij} + w_{ik})}{2} a_{ij}.a_{ik}.a_{jk} \tag{3-4}$$

where $D(v_i)$ is the number of nodes adjacent to node v_i, and a_{ij} is 1 if node v_i is adjacent to node v_j, and 0 otherwise. $SS(v_i)$ is the strength of node v_i and w_{ij} is the random variable associated with the weight of edge e_{ij}.

3.2. Some algorithms for computing network measures in stochastic graphs

In the previous section, we defined some network measures for stochastic graphs. In this section, several algorithms based on learning automata are introduced for computing network measures in stochastic graphs under the situation that the probability distribution functions of the weights associated with the edges of graph are unknown. The algorithms for computing network measures by taking samples from the edges of the stochastic graph try to estimate the distribution of the network measures. The process of sampling from the edges of the graph is guided by the aid of learning automata in such a way that the number of samples needed to be taken from the edges of the stochastic graph for estimating the network measures to be reduced as much as possible. In the algorithms, the guided sampling process implemented by learning automata aims to take more samples from the promising region of the graph, the regions that reflects higher rate of activities, instead of walking around and taking unnecessary samples from non-promising region of the graph. The algorithm after initialization phase iterates the updating phase until one of the stopping conditions is met. The details of initialization phase, updating phase, and stopping conditions are given below.

3.2.1. Initialization

Let $G\langle V, E, W\rangle$ be the input stochastic graph, where $V=\{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $E=\{e_{i,j}\}\subseteq V\times V$ is the edge-set, and $W=\{w_{i,j}\}$ is the set of random variables each of which is associated with an edge weight of the input stochastic graph. It is assumed that the weight of each edge is a positive random variable with unknown probability distribution function. The

algorithm uses a network of learning automata isomorphic to the input graph and tries to estimate weights of edges by sampling from the edges of the graph in such a way that the number of samples taken from the edges of graph be reduced. Learning automaton A_i residing in node v_i has two actions as $\alpha_i = \{\alpha^1, \alpha^2\}$ where action α^1 is "take sample from the edges of chosen node v" and action α^2 is "do not take sample from the edges of chosen node v". Let $p(v_i) = \{p_i^1, p_i^2\}$ be the action probability vector of learning automaton A_i and initialized equally $p_i^1 = p_i^2 = 1/2$ for all $v_i \in V$. The process of taking samples from the edges on which a node is incident is governed by the learning automaton assigned to that node. At the beginning of the algorithm, the weight of each edge is initialized with some random samples in order to provide a coarse estimate of the weight of that edge.

3.2.2. **Updating**

In this phase, every learning automaton in the network chooses an action according to its action probability vector in parallel manner. Then a sample from every edge incident on a node whose corresponding learning automaton has chosen the action of taking sample (α^1) will be taken. Computation of the new estimates for the means and standard deviations of the unknown distributions of the edge weights are then performed. Using these new estimates, the network measure is computed for the given network and then the difference between the computed network measure and the network measure obtained in the previous iteration is computed using DDQC (equation (3-11)). Based on this difference, the probability vectors of learning automata assigned to the nodes of the input graph are updated according to the learning algorithm. That is, if this difference is lower than or equal to a given error ε_p, then action α^1 of all learning automata are rewarded proportional to the amount of improvement and penalized otherwise.

3.2.3. **Stopping**

Updating phase is repeated until one of the following stopping conditions is met.

- The number of iterations k exceeds a given number K_{max},
- The difference between the computed measure in two consecutive iterations becomes lower than a specified error E_{min}, or
- The average of entropy of probability vector of learning automata reaches a predefined value T_{min}.

In the algorithm, the entropy is used for measuring learning process. The information entropy for a learning automaton with r actions can be defined as follows (A. Mousavian, Rezvanian, and Meybodi 2013):

$$H = -\sum_i^r p_i . Log(p_i) \qquad (3\text{-}5)$$

where p_i is the probability of choosing i^{th} action of a learning automaton. The entropy for a learning automaton has maximum value of one when all the actions have equal

probabilities of choosing and has minimum value of zero when the action probability vector is a unit vector. The pseudo-code of algorithm 1 is shown in Figure 3-1.

Algorithms 3-1. The algorithm for computing network measures in stochastic graphs

Input: Stochastic Graph $G\langle V, E, W\rangle$, Network measure M_Θ, Thresholds K_{max}, T_{min}, E_{min}, ε_p

Output: Computed network measures $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n$ where $\hat{\theta}_i$ is the computed network measure for node v_i

// That is $\hat{\theta}_i$ is the expected of ES_i (Eq. (3-1) if the given network measure M_Θ is the strength of the network,

// $\hat{\theta}_i$ is the expected of EC_i (Eq. (3-2) if the given network measure M_Θ is the closeness of the network,

// $\hat{\theta}_i$ is the expected of EB_i (Eq. (3-3) if the given network measure M_Θ is the betweenness of the network, and

// $\hat{\theta}_i$ is the expected of ECC_i (Eq. (3-4) if the given network measure M_Θ is the clustering coefficient of the network.

Initialization:

Create a network of learning automata isomorphic to the graph G by assigning an automaton A_i to each node v_i;

Let $\alpha_i = \{\alpha^1, \alpha^2\}$ be the set of actions for learning automata A_i;

Let $P=\{p_1, p_2, \ldots, p_n\}$ be the set of action probabilities in which $p_i=\{p_i^1, p_i^2\}$ and initialized equally $p_i^1 = p_i^2 = 1/2$ for all $v_i \in V$;

Let k be the iteration number and initially set to 1;

Let $\bar{w}_{i,j}$ be the estimate for mean of edge weight of all samples taken from edge e_{ij} and initialized with some random samples;

Let $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ be the computed network measure for the nodes at iteration k;

Let G^k be the network with mean edge weight at iteration k;

Begin

Let $H_i(k)$ be the entropy of learning automaton A_i at iteration k and computed based on equation (3-5);

Repeat

For all learning automata do in parallel

Learning automaton A_i chooses an action according to its action probability vector;

If ($\alpha_i=\alpha^1$) **then**

Take a sample from every edge on which node v_i is incident;

Construct G^k by computing the new weight for every edge on which node v_i is incident;

Else

No action;

End if

End for

If (M_Θ is strength**)**

Compute new estimates for the strengths of the nodes $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-1);

Else If (M_Θ is closeness)

Compute new estimates for the closeness $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-2);

Else If (M_Θ is betweenness)

Compute new estimates for the betweenness $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-3);

Else If (M_Θ is clustering coefficient)

Compute new estimates for the clustering coefficient $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-4);

End if

Compute $DDQC^k(G^k, G^{k-1})$ using equation (3-11); // DDQC between consequent estimated network measures

If ($k > 1$) **Then**

Compute $\Delta^k = DDQC^k - DDQC^{k-1}$; // Compute the change rate between iterations k and k-1;

If ($|\Delta^k| \geq \varepsilon_p$) **Then**

Reward the chosen actions of all learning automata;

Else

Penalize the chosen actions of all learning automata;

End If

End If

$H(k) \leftarrow \frac{1}{n}\sum_{i=1}^{n} H_i(k)$;
$k \leftarrow k + 1$;
Until ($k < K_{max}$ **OR** $H(k) > T_{min}$ **OR** $|\Delta^k| > E_{min}$)
End Algorithm

Figure 3-1. Pseudo-code for algorithm 1 for compute network measures in stochastic graphs.

3.3. **Improvements**

In the remaining part of this section some improvements of Algorithm 1 are described. These modifications try to improve either the speed or the efficiency of the algorithm or both of them.

3.3.1. **Improvement I**

The performance of any learning automata based algorithm is very sensitive to learning rate of the learning algorithm. Large value for learning rate results in increasing the speed of convergence and decreasing the accuracy of the algorithm while small value for learning rate results in increasing the accuracy and decreasing the speed of convergence of the algorithm. In Algorithm 1, all learning automata use a same learning rate proportional to the amount of improvement. It means that the algorithm in each iteration treats all nodes equally during the execution of the algorithm. For this improvement, we use $a_i(k) = c_k.d_i^j(k)$ as the learning parameter of automaton A_i at iteration k where $c_k = \Delta^k/Max(\Delta^k)$ is a scaled change rate between 0 and 1 at iteration k, $d_i^j(k)$ is an estimation of reward probability for action α^j computed by the algorithm at iteration k. $d_i^j(k)$ is computed by dividing the number of times that action α^j is chosen up to time k by automaton A_i and received reward by the number of times that automaton A_i is activated. In this way, the edges which were chosen more frequently will be more probable to be chosen in the future. As in the experiment, such a strategy will lead to higher rate of convergence and fewer samples from the graph. Algorithm 1 in which the learning rate is adapted according to this improvement is called *Algorithm 2*.

3.3.2. **Improvement II**

For this improvement, we use the information entropy of a learning automaton (defined by equation (3-5) for adaptation of learning rate of that learning automaton. The learning rate of learning automaton A_i at iteration k is computed as $a_i(k) = c_k.H_i(k)$ where $c_k = \Delta^k / Max(\Delta^k)$ is a scaled change rate between 0 and 1 at iteration k and $H_i(k)$ is the information entropy of learning automaton A_i which reflect the uncertainty of A_i about their actions. In this way, the nodes whose estimated edge weights have more changes will be more probable to be chosen in the future for the sampling purpose. The results of experiment 5 show that such a strategy will lead to a higher rate of convergence and also fewer samples taken from the graph. Algorithm 1 in which the learning rate is adapted using the information entropy is called *Algorithm 3*.

3.3.3. **Improvement III**

In algorithm 1, we may reach a point that the changes in probability action vectors of some of the learning automata in two consecutive iterations are negligible which in this case we may turn off these learning automata in order to prevent taking more samples from the edges which may not lead us to a much better estimation of the edges weights and consequently better estimate of network measures. Algorithm 1 in which the above improvement is applied is called *Algorithm 4*. Experiment 5 shows that algorithm 4 comparing to algorithm 1 produces higher convergence speed for all test graphs.

3.3.4. **Improvement IV**

This improvement is obtained by equipping each node with two learning automata, the first learning automata as in algorithm 1 decides whether or not a node to be chosen and the second learning automata (with number of actions equal to the number of nodes adjacent to that node) decides which edge on which the node is incident must be sampled. Experimentation shows that the utilization of another automaton in each node will lead to a higher rate of convergence and also taking fewer samples from the edges of the graph. Algorithm 1 in which the above improvement is used is called *Algorithm 5*. Experiment 2 shows that algorithm 5 produces fewer numbers of samples and higher accuracy taken from the graph as compared to the case where the second learning automata are absent.

3.3.5. **Improvement V**

This improvement is obtained by combing improvements 2, 3 and 4. The resulting algorithm is called *Algorithm 6*. The pseudo-code of this algorithm is given in Figure 3-2.

Algorithms 3-2. The algorithm for Computing network measures in stochastic graphs

Input: Stochastic Graph $G\langle V, E, W\rangle$, Network measure M_Θ, Thresholds K_{max}, T_{min}, E_{min}, ε_h, ε_p

Output: Computed network measures $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n$ where $\hat{\theta}_i$ is the computed network measure for node v_i

// That is $\hat{\theta}_i$ is the expected of ES_i (Eq. (3-1)) if the given network measure M_Θ is the strength of the network,

// $\hat{\theta}_i$ is the expected of EC_i (Eq. (3-2) if the given network measure M_Θ is the closeness of the network,

// $\hat{\theta}_i$ is the expected of EB_i (Eq. (3-3) if the given network measure M_Θ is the betweenness of the network, and

// $\hat{\theta}_i$ is the expected of ECC_i (Eq. (3-4) if the given network measure M_Θ is the clustering coefficient of the network.

Initialization:

Create the first network of learning automata A_1 isomorphic to the graph G by assigning an automaton A_1^i to each node v_i;

Create the second network of learning automata A_2 isomorphic to graph G by assigning an automaton A_2^i to each node v_i;

Let $\alpha_1=\{\alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,n}\}$ be the set of actions for the first network of learning automata and $\alpha_{1,i} = \{\alpha_{1,i}^1, \alpha_{1,i}^2\}$ is the set of actions for learning automaton A_1^i;

Let $\alpha_2=\{\alpha_{2,1}, \alpha_{2,2}, \ldots, \alpha_{2,n}\}$ be the set of actions for the second network of learning automata and $\alpha_{2,i} = \{\alpha_{2,i}^1, \alpha_{2,i}^2, \ldots, \alpha_{2,i}^{ri}\}$ is the set of actions for learning automaton A_2^i with $r_i=|v_i|$ actions;

Let $P_1=\{p_{1,1}, p_{1,2}, \ldots, p_{1,n}\}$ be the set of action probabilities of the first network of learning automata in which $p_{1,i}=\{p_{1,i}^1, p_{1,i}^2\}$ and initialized equally $p_{1,i}^1 = p_{1,i}^2=1/2$ for all $v_i \in V$;

Let $P_2=\{p_{2,1}, p_{2,2}, \ldots, p_{2,n}\}$ be the set of action probabilities of the second network of learning automata in which $p_{2,i}=\{p_{2,i}^1, p_{2,i}^2, \ldots, p_{2,i}^{ri}\}$ and initialized equally $p_{2,i}^1 = p_{2,i}^2 = \cdots = p_{2,i}^{ri}=1/r_i$ for all $v_i \in V$;

Let S_i be the state of learning automaton A_1^i and initially set to active for all learning automata A_1;

Let k be the iteration number and initially set to 1;

Let $\overline{w}_{i,j}$ be the estimate for mean of edge weights of all samples taken from edge e_{ij} and initialized with some random samples;

Let $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ be the computed network measure for the nodes at iteration k;

Let G^k be the network with mean edge weight at iteration k;

Begin

Let $H_i(k)$ be the information entropy of learning automaton A_1^i at iteration k and computed based on equation (3-5);

Repeat

For all active first learning automata do in parallel

Each learning automaton A_1^i randomly choose an action according to its action probability vector;

If ($\alpha_{1,i}$=$\alpha_{1,i}^1$) **then**

Learning automaton A_2^i randomly chooses an action according to its action probability vector;

Let the chosen action be $\alpha_{2,i}^j$;

Take a sample from edge $e(v_i, v_j)$;

Construct G^k by computing the new estimate for weight of edge $e(v_i, v_j)$;

Else

No action;

End if

If ($k > 1$ **AND** $H_i(k) > T_{min}$ **AND** $|H_i(k)-H_i(k-1)| < \varepsilon_h$)

$S_i \leftarrow 0$; // turn off learning automaton A_1^i

End if

End for

If (M_Θ is strength)

Compute new estimates for the strengths of the nodes $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-1);

Else If (M_Θ is closeness)

Compute new estimates for the closeness $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-2);

Else If (M_Θ is betweenness)

Compute new estimates for the betweenness $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-3);

Else If (M_Θ is clustering coefficient)

Compute new estimates for the clustering coefficient $\hat{\theta}_1^k, \hat{\theta}_2^k, \ldots, \hat{\theta}_n^k$ using equation (3-4);

End if

Compute $DDQC^k(G^k, G^{k-1})$ using equation (3-11); // DDQC between consequent estimated network measures

If ($k > 1$) **Then**

Compute $\Delta^k \leftarrow DDQC^k - DDQC^{k-1}$; // the change rate between iterations k and k-1;

Compute $c_k \leftarrow \frac{\Delta^k}{Max(\Delta^k)}$; // normalized coefficient;

Compute $H(k) \leftarrow \frac{1}{n}\sum_{i=1}^{n} H_i(P)$; // information entropy;

$a_i(k) \leftarrow c_k . H_i(k)$; // update learning parameter;

If ($|\Delta^k| \geq \varepsilon_p$) **Then**

Reward the chosen actions of both learning automata according to learning algorithm;

Else

Penalize the chosen actions of both learning automata according to learning algorithm;

End If

End If

$k \leftarrow k + 1$;

Until ($k < K_{max}$ **OR** $H(k) > T_{min}$ **OR** $|\Delta^k| > E_{min}$)

End Algorithm

Figure 3-2. Pseudo-code for algorithm 6 for computing network measures in stochastic graphs.

3.4. **Experiments**

In this section, to study the performance of the algorithms, several computer simulations are conducted on synthetic stochastic graphs which are generated based on well-known random graph with their weights of edges being random variables. We use well-known computer generated network model of *Barabási-Albert* model (*BA* model) as a scale-free network with heavy-tailed degree distribution (Barabási and Albert 1999) which is utilized widely in literature. We set network parameters of BA model as $N \in \{1000, 2000, 5000, 10000\}$ and $m_0 = m = 5$. Other synthetic network as a small world network is *Watts–Strogatz* model (WS model) (Watts and Strogatz 1998) which reflect a common property of many real networks such as short average path length with $N \in \{1000, 2000, 5000, 10000\}$, $k = 4$ and p=0.2. Also, we use well-known random network of *Erdös–Rényi* model (*ER* model) (Erdos and Rényi 1960) which is utilized widely in literature as $N \in \{1000, 2000, 5000, 10000\}$ and p=0.2. The edge weights for these graphs are random variables with exponential distributions whose mean are chosen randomly from set $\mu \in \{0.8, 1, 1.2, 1.5\}$. It is noted that the chosen mean values is adopted from an empirical observations by *Bild et al* (Bild et al. 2015).

3.4.1. **Evaluation measures**

In order to evaluate the accuracy of estimations by the algorithms, we used several distance function between real parameters and estimated parameters. In this chapter, we use Kolmogorov-Smirnov distance statistic, skew divergence distance, Pearson's correlation coefficient and normalized L_1 distance as distance measures to compare distribution properties of the algorithms for estimating network parameters. These distance measures as evaluating criteria are described below.

3.4.1.1. **Kolmogorov-Smirnov D-statistics (KS-D)**

Kolmogorov-Smirnov (KS) D-statistics is one of the statistical test methods commonly used for assessment the distance between two cumulative distribution functions (CDFs). *KS* measures acceptability between original distribution and estimated distribution. The result of this test is a value between 0 and 1 that as closer as it is to zero, both distributions will have a greater similarity; and as closer as it is to unit, the two distributions will show a greater discrepancy. This measure has been defined as

$$KS(P, Q) = \max_x |P(x) - Q(x)| \tag{3-6}$$

where P and Q are two CDFs of original and estimated data, respectively, and x represents the range of the random variable. So it is computed as the maximum vertical distance between the two distributions.

3.4.1.2. **Skew divergence (SD)**

Skew divergence can also be used for assessment the distance between two Probability Distribution Functions (PDF) and defined as follows

$$SD(P,Q,\alpha) = D[\alpha P + (1-\alpha)Q || \alpha Q + (1-\alpha)P] \quad (3\text{-}7)$$

where D is the *Kullback-Leibler* (KL) *divergence*, which measures the similarity between two PDFs P and Q that do not have continuous support over the full range of values and α=0.99. The KL *divergence* is defined as follows

$$KL(P(x)||Q(x)) = \sum_x P(x) log \frac{P(x)}{Q(x)} \quad (3\text{-}8)$$

3.4.1.3. **Pearson's correlation coefficient (PCC)**

One of the good indices to measure the similarity between the estimated parameter values and the original parameters values is Pearson's correlation coefficient (PCC) (Luo et al. 2015) and Computed as follows

$$PCC(P,Q) = \frac{n\sum_i p_i q_i - (\sum_i p_i)(\sum_i q_i)}{\sqrt{n\sum_i p_i^2 - (\sum_i p_i)^2}\sqrt{n\sum_i q_i^2 - (\sum_i q_i)^2}} \quad (3\text{-}9)$$

where p_i and q_i are the values of the original parameter P and estimated parameter Q in the original data and obtained samples data respectively and also n is the number of elements for each parameters. If the result of PCC is much closer to 1, then the similarity of real and estimated parameters is much better.

3.4.1.4. **Normalized L_1 distance**

In some cases, for evaluation, the distance between two positive m-dimensional real vectors P and Q are measured where P is the original vector and Q is the estimated vector. Normalized L_1 distance (Ahmed, Neville, and Kompella 2014) is defined as follows

$$L_1(P,Q) = \frac{1}{n}\sum_i \frac{|p_i - q_i|}{p_i} \quad (3\text{-}10)$$

3.4.1.5. **Degree Distribution Quantification and Comparison (DDQC)**

In this chapter, in the algorithm, DDQC measures the difference between two computed network measures in the consecutive iterations. Recently, DDQC, as a new method for quantification and comparison of network degree distributions is presented (Aliakbary, Habibi, and Movaghar 2015). This method is an improved version of *Janssen*'s method (Janssen, Hurshman, and Kalyaniwalla 2012) for feature extraction from the degree distribution. The DDQC can be applied for any network measure distribution. In DDQC, the given degree distribution divided into eight regions based on the following description and the sum of parameter distribution in each region is computed as distribution percentiles. The absolute difference between these extracted values for real and estimated distributions in the original data and obtained samples data is considered as the distance between two distributions as given following equation.

$$DDQC(G,G') = \sum_{i=1}^{8} |Q_i(G) - Q_i(G')| \tag{3-11}$$

where $Q(G)=\langle IDP(I(i))\rangle_{i=1,\dots,8}$ is the set of values for eight regions of graph G and $IDP(I)$ is the interval distribution probability and can be computed as given below

$$IDP(I) = Pr(left(I) \leq D \leq right(I)) \tag{3-12}$$

where *Left(R)* and *Right(R)* is the minimum and maximum values for each interval *I*. The interval *I* can be divided in two regions as following

$$I(i) = \begin{cases} \left[left\left(R\left(\left\lceil\frac{i}{2}\right\rceil\right)\right), left\left(R\left(\left\lceil\frac{i}{2}\right\rceil\right)\right) + \frac{\left|R\left(\left\lceil\frac{i}{2}\right\rceil\right)\right|}{2}\right] & i\ is\ odd \\ \left[left\left(R\left(\left\lceil\frac{i}{2}\right\rceil\right)\right) + \frac{\left|R\left(\left\lceil\frac{i}{2}\right\rceil\right)\right|}{2}, right\left(R\left(\left\lceil\frac{i}{2}\right\rceil\right)\right)\right] & i\ is\ even \end{cases} \tag{3-13}$$

where

$$|R(r)| = \max\big(right\big(R(r)\big) - left\big(R(r)\big), 0\big) \tag{3-14}$$

is the range length of values. The range of values is divided in four regions based on mean and standard deviation of a network measure.

$$R(r) = \begin{cases} [D_{min}, \mu - \sigma] & r = 1 \\ [\mu - \sigma, \mu] & r = 1 \\ [\mu, \mu + \sigma] & r = 1 \\ [\mu + \sigma, D_{max}] & r = 1 \end{cases} \tag{3-15}$$

where $\mu = \sum_{D_{min}}^{D_{max}} x \times Pr(x)$ and $\sigma^2 = \sum_{D_{min}}^{D_{max}} Pr(x) \times (x-\mu)^2$ are the mean and variance of the probability distribution function *Pr(x)* respectively, which obtained from a network measure.

3.4.2. **Experimental setup**

The stopping criteria for the algorithms is either the number of iteration k reached $K_{max}=50\times n$ iterations where n is the number of nodes for each instance graph or the value of average information entropy $H(k)$ is lower than $T_{min}=0.10$ or the difference between the computed measure in two consecutive iteration Δ^k becomes lower than $\varepsilon_h=0.001$. The edge weights for synthetic stochastic graphs are random variables with exponential distributions whose mean are selected randomly from set $\mu\in\{0.8, 1, 1.2, 1.5\}$. For all algorithms, the reinforcement scheme used for updating the action probability vector of learning automata is $L_{RεP}$ with $a=0.05$, $b=0.01$ and $a=0.01$, $b=0.005$ for the first and the second set of learning automata, respectively. For algorithm 4, a learning automation is disabled when its information entropy reaches 0.05. The reported results are the averages taken over 30 runs.

In this book, we assume that the structural and behavioral parameters of online social networks are time varying parameters which can be observed by the network. Using the observed parameters, the network can compute different measurements such as strength, closeness and betweenness in an ongoing fashion. The algorithm introduced in section 4 can be used by the network as a means for observing the time varying parameters of the network for the purpose of network's measurements computations. The aim of this algorithm is to collect information from the network in order to find good estimates for the network's measurements using fewer numbers of samples than that of standard sampling methods. The first experiment that follows gives a hint how one can use the approach introduced in the chapter to analyze a social network modeled as a stochastic graph. The remaining experiments (experiments 2 through 5) study the performance of the algorithms introduced for estimation of the network's measurements.

3.4.3. **Experimental results**

3.4.3.1. **Experiment I**

Using the distributions of the measures for the nodes one can analyze the user behavior and human activities of an online social network using some statistical tests. The following experiments elaborate on this. In this experiment, we select a set of six randomly nodes from synthetic stochastic BA graphs and then ranked them according to their strength, closeness, betweenness and clustering coefficient. The distributions of betweenness, clustering coefficient, closeness and strength for these randomly selected nodes are given in Figure 3-3 through Figure 3-6, respectively. To obtain the statistical ranking, we use multi-comparison statistical tests using Friedman (García et al. 2009) with 95% significance interval (α=0.05). The average ranking for multi-comparison statistical tests using Friedman's test are presented in Table 3-1 and the results of *p*-values of the comparing nodes in terms of different network measures are given in Table 3-2 through Table 3-5. The *p*-value computed by Friedman's test is 2.34E-06, which is below the significance interval of 95% (α=0.05). Thus, a significant difference exists among the observed results. In applications such as rumor spreading, disease propagation, and protecting critical regions from intended attacks we need to study the problem of maximization of spread of influence which involves ranking the set of nodes in the network according to their strengths or closeness centralities (Li et al. 2014).

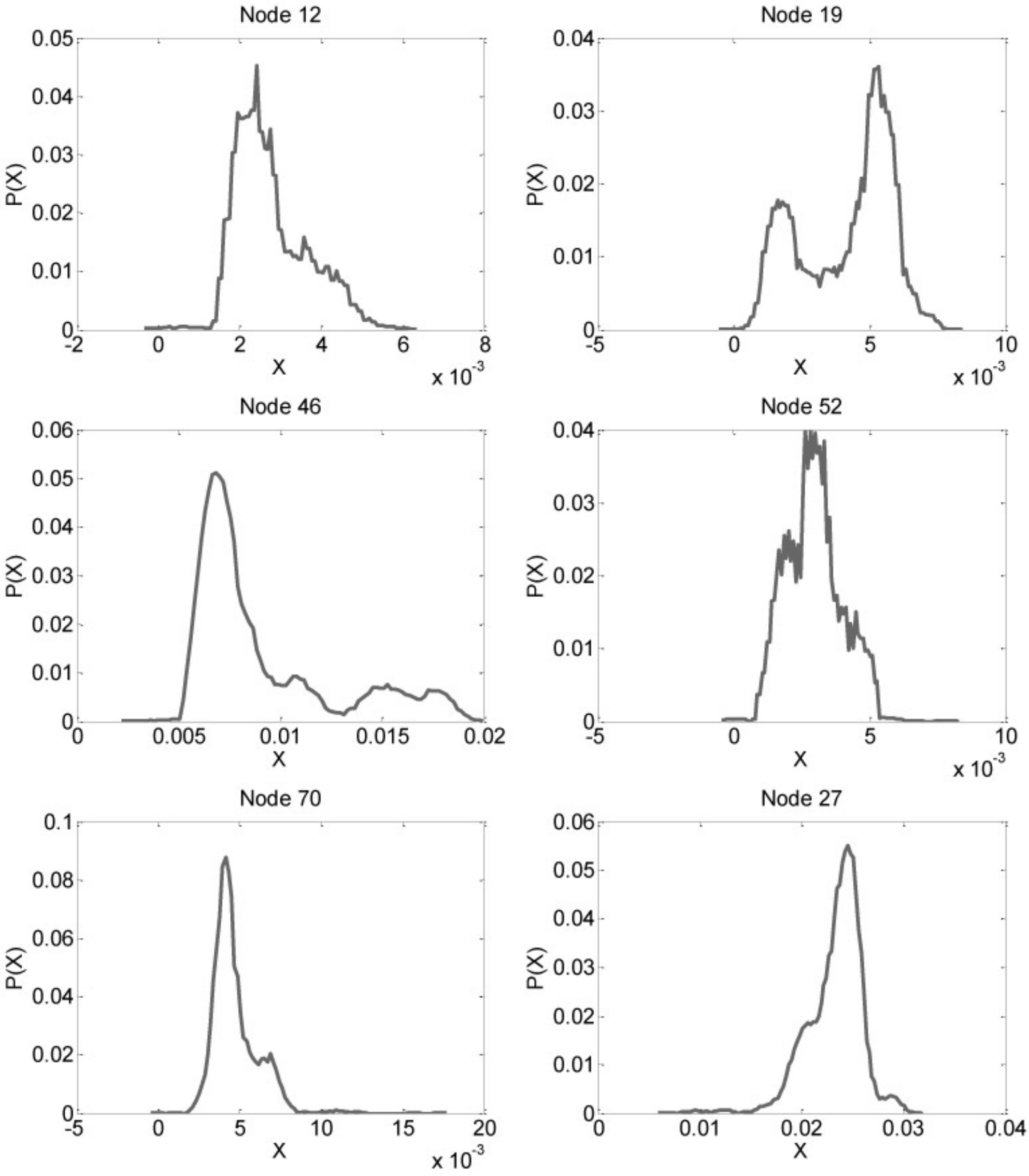


Figure 3-3. Distributions of betweenness for some randomly selected nodes of the synthetic stochastic BA network.

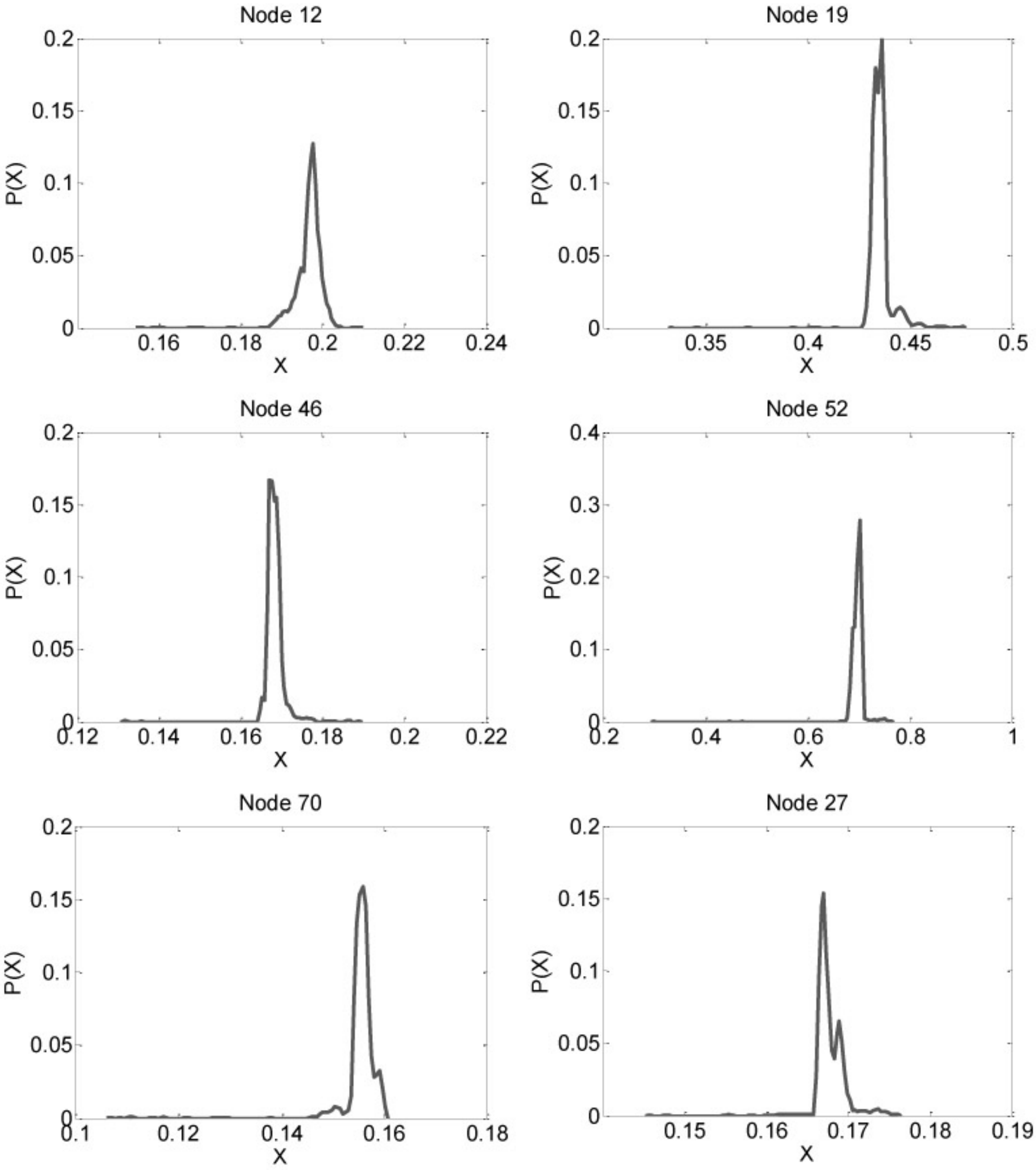


Figure 3-4. Distributions of clustering coefficient for some randomly selected nodes of the synthetic stochastic BA network.

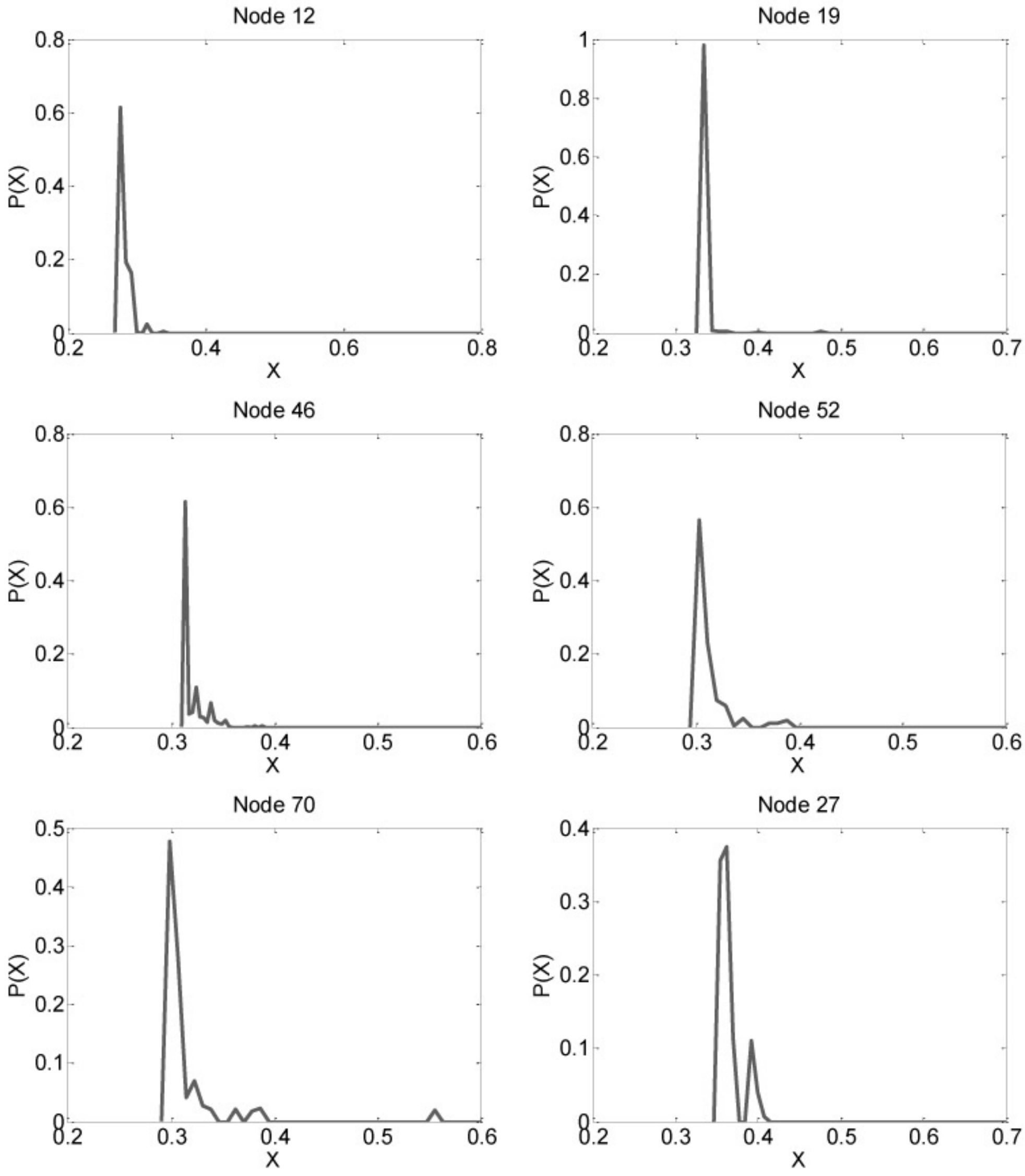


Figure 3-5. Distributions of closeness for some randomly selected nodes of the synthetic stochastic BA network.

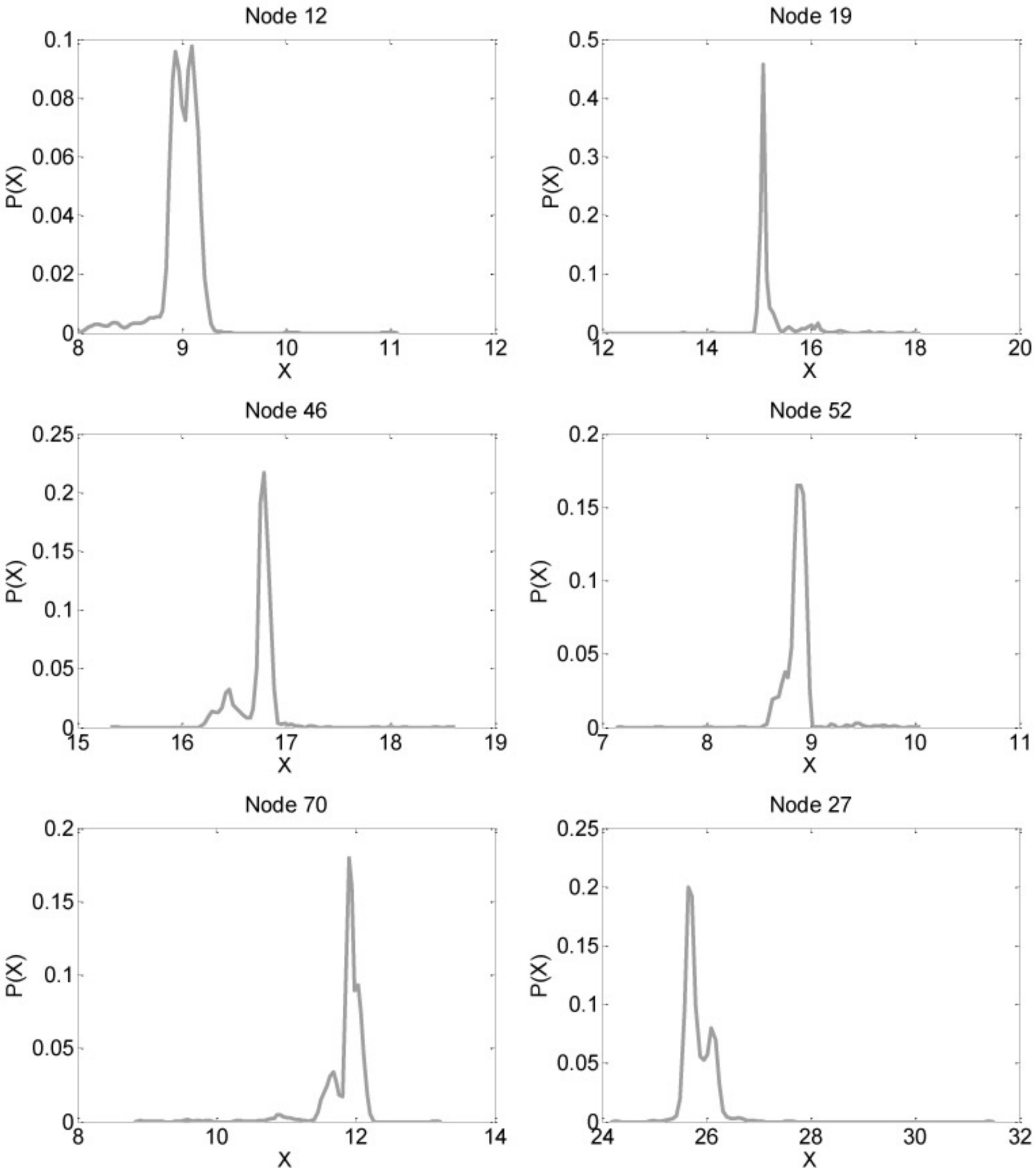


Figure 3-6. Distributions of strength for some randomly selected nodes of the synthetic stochastic BA network.

Table 3-1. Average ranking of Friedman's test for randomly selected nodes of the synthetic stochastic BA network.

	Betweenness		Clustering coefficient		Closeness		Strength	
Randomly selected nodes	Average Ranking	Ranking	Average Ranking	Ranking	Average Ranking	Ranking	Average Ranking	Ranking
Node 12	5.98	6	1.99	2	2.18	2	2.98	3
Node 19	1.49	1	4.64	5	1.02	1	1.00	1
Node 27	2.70	2	4.34	4	2.90	3	2.01	2
Node 46	4.83	5	1.01	1	4.03	4	5.84	6

Node 52	2.97	3	6.00	6	4.86	5	4.01	4
Node 70	3.01	4	3.01	3	5.99	6	5.15	5

Table 3-2. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of betweenness of nodes.

i	Hypothesis	Unadjusted *p*-value	*Holm p*-value	*Shaffer p*-value
15	Node 12 vs. Node 19	0.00E+00	3.33E-03	3.33E-03
14	Node 19 vs. Node 46	0.00E+00	3.57E-03	5.00E-03
13	Node 12 vs. Node 27	0.00E+00	3.85E-03	5.00E-03
12	Node 12 vs. Node 52	0.00E+00	4.17E-03	5.00E-03
11	Node 12 vs. Node 70	0.00E+00	4.55E-03	5.00E-03
10	Node 27 vs. Node 46	0.00E+00	5.00E-03	5.00E-03
9	Node 46 vs. Node 52	0.00E+00	5.56E-03	7.14E-03
8	Node 46 vs. Node 70	0.00E+00	6.25E-03	7.14E-03
7	Node 19 vs. Node 70	0.00E+00	7.14E-03	7.14E-03
6	Node 19 vs. Node 52	0.00E+00	8.33E-03	8.33E-03
5	Node 19 vs. Node 27	0.00E+00	1.00E-02	1.25E-02
4	Node 12 vs. Node 46	0.00E+00	1.25E-02	1.25E-02
3	Node 27 vs. Node 70	1.12E-02	1.67E-02	1.67E-02
2	Node 27 vs. Node 52	2.20E-02	2.50E-02	2.50E-02
1	Node 52 vs. Node 70	8.06E-01	5.00E-02	2.50E-02

Note: Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 5.00E-02; Shaffer's procedure rejects those hypotheses that have an unadjusted p-value ≤3.33E-03.

Table 3-3. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of clustering coefficient of nodes.

i	Hypothesis	Unadjusted *p*-value	*Holm p*-value	*Shaffer p*-value
15	Node 46 vs. Node 52	0.00E+00	3.33E-03	3.33E-03
14	Node 12 vs. Node 52	0.00E+00	3.57E-03	5.00E-03
13	Node 19 vs. Node 46	0.00E+00	3.85E-03	5.00E-03
12	Node 27 vs. Node 46	0.00E+00	4.17E-03	5.00E-03
11	Node 52 vs. Node 70	0.00E+00	4.55E-03	5.00E-03
10	Node 12 vs. Node 19	0.00E+00	5.00E-03	5.00E-03
9	Node 12 vs. Node 27	0.00E+00	5.56E-03	7.14E-03
8	Node 46 vs. Node 70	0.00E+00	6.25E-03	7.14E-03
7	Node 27 vs. Node 52	0.00E+00	7.14E-03	7.14E-03
6	Node 19 vs. Node 70	0.00E+00	8.33E-03	8.33E-03
5	Node 19 vs. Node 52	0.00E+00	1.00E-02	1.25E-02
4	Node 27 vs. Node 70	0.00E+00	1.25E-02	1.25E-02
3	Node 12 vs. Node 70	0.00E+00	1.67E-02	1.67E-02
2	Node 12 vs. Node 46	0.00E+00	2.50E-02	2.50E-02
1	Node 19 vs. Node 27	9.24E-03	5.00E-02	5.00E-02

Note: Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 3.33E-03; Shaffer's procedure rejects those hypotheses that have an unadjusted p-value ≤3.33E-03.

Table 3-4. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of closeness of nodes.

i	Hypothesis	Unadjusted *p*-value	*Holm p*-value	*Shaffer p*-value
15	Node 19 vs. Node 70	0.00E+00	3.33E-03	3.33E-03
14	Node 19 vs. Node 52	0.00E+00	3.57E-03	5.00E-03

13	Node 12 vs. Node 70	0.00E+00	3.85E-03	5.00E-03
12	Node 27 vs. Node 70	0.00E+00	4.17E-03	5.00E-03
11	Node 19 vs. Node 46	0.00E+00	4.55E-03	5.00E-03
10	Node 12 vs. Node 52	0.00E+00	5.00E-03	5.00E-03
9	Node 27 vs. Node 52	0.00E+00	5.56E-03	7.14E-03
8	Node 46 vs. Node 70	0.00E+00	6.25E-03	7.14E-03
7	Node 19 vs. Node 27	0.00E+00	7.14E-03	7.14E-03
6	Node 12 vs. Node 46	0.00E+00	8.33E-03	8.33E-03
5	Node 12 vs. Node 19	0.00E+00	1.00E-02	1.25E-02
4	Node 27 vs. Node 46	0.00E+00	1.25E-02	1.25E-02
3	Node 52 vs. Node 70	0.00E+00	1.67E-02	1.67E-02
2	Node 46 vs. Node 52	0.00E+00	2.50E-02	2.50E-02
1	Node 12 vs. Node 27	0.00E+00	5.00E-02	5.00E-02

Note: Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 3.33E-03; Shaffer's procedure rejects those hypotheses that have an unadjusted p-value ≤3.33E-03.

Table 3-5. *p*-values of the comparing nodes on the synthetic stochastic BA networks in terms of strength of nodes.

i	Hypothesis	Unadjusted *p*-value	*Holm p*-value	*Shaffer p*-value
15	Node 19 vs. Node 46	0.00E+00	3.33E-03	3.33E-03
14	Node 19 vs. Node 70	0.00E+00	3.57E-03	5.00E-03
13	Node 27 vs. Node 46	0.00E+00	3.85E-03	5.00E-03
12	Node 27 vs. Node 70	0.00E+00	4.17E-03	5.00E-03
11	Node 19 vs. Node 52	0.00E+00	4.55E-03	5.00E-03
10	Node 12 vs. Node 46	0.00E+00	5.00E-03	5.00E-03
9	Node 12 vs. Node 70	0.00E+00	5.56E-03	7.14E-03
8	Node 27 vs. Node 52	0.00E+00	6.25E-03	7.14E-03
7	Node 12 vs. Node 19	0.00E+00	7.14E-03	7.14E-03
6	Node 46 vs. Node 52	0.00E+00	8.33E-03	8.33E-03
5	Node 52 vs. Node 70	0.00E+00	1.00E-02	1.25E-02
4	Node 12 vs. Node 52	0.00E+00	1.25E-02	1.25E-02
3	Node 19 vs. Node 27	0.00E+00	1.67E-02	1.67E-02
2	Node 12 vs. Node 27	0.00E+00	2.50E-02	2.50E-02
1	Node 46 vs. Node 70	0.00E+00	5.00E-02	5.00E-02

Note: Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 3.33E-03; Shaffer's procedure rejects those hypotheses that have an unadjusted p-value ≤3.33E-03.

3.4.3.2. **Experiment II**

In this experiment, we compare the number of samples required by the algorithms in order to reach certain accuracy with the number of samples needed by the standard sampling method (SSM) to reach the same accuracy. In order to have a fair comparison, both the algorithm and SSM must use the same confidence level 1-ε. The algorithm may reach the confidence level of 1-ε by a proper choice of learning parameter a. According to appendix C, such learning parameter can be estimated using $ax/(e^{ax}-1) = max_{i\neq j}(d_j/d_i)$ where d_i is the reward probability of action α_i. Based on the SSM, to obtain a certain confidence

level 1-ε for a subset of edges in stochastic graph, we need to build a confidence level 1-ε_i for each edge e_i such that $\varepsilon = \sum_{i=1}^{m} \varepsilon_i / m$. In this experiment, same confidence level 1-ε_0 is assumed for all edges of the stochastic graph. The minimum required number of samples for each edge of graph for SSM is computed subject to $p[|\bar{x}_n - \mu| < \delta] \geq 1 - \varepsilon$, where δ=0.01.

For this experimentation, error rate ε varies from 0.3 to 0.05 with increment 0.05 (confidence levels 0.7 to 0.95) for the algorithms. Different synthetic stochastic graphs with size from 1000 to 10000 are used. The results of this experimentation are averages taken over all the test networks with respect to edge sampling rate (ESR) which is defined as the average number of samples that is needed to be taken from each edge of stochastic graphs to reach a certain confidence interval (CI). The average and standard deviation of ESR are given in Table 3-6 through Table 3-8 for synthetic stochastic ER, BA and WS graphs, respectively. In order to show the significant statistical difference among the algorithms, a non-parametric statistical test called Wilcoxon rank sum test is conducted for independent samples (García et al. 2009; Wilcoxon 1945) at the 0.05 significance level. In Table 3-6 through Table 3-8, the test result regarding corresponding Algorithm versus Algorithm 6 is shown as "†", "§" and "≈" when corresponding algorithm is better than, worse than, and similar to that of Algorithm 6, respectively. From the results, several conclusions can be made: 1) For all the algorithms, the average number of samples taken from each edge of graph is fewer than the number of samples taken using SSM for all synthetic stochastic graphs and different confidence levels; 2) From the results, we may observe that Algorithm 6 requires the lowest number of samples from each edge as compared to algorithms 1 through 4 for all synthetic stochastic graphs and different confidence levels.

Table 3-6. Average results [± standard deviation (std)] for SSM and the algorithms for synthetic stochastic ER graphs with different error rates in terms of ESR.

Methods \ CI	0.7	0.75	0.8	0.85	0.9	0.95
SSM	24.54 ± 0.77§	30.00 ± 1.08§	36.98 ± 1.34§	46.42 ± 1.69§	60.26 ± 2.20§	85.15 ± 3.13§
Algorithm 1	9.49 ± 2.82§	12.66 ± 3.62§	17.46 ± 3.06§	24.50 ± 5.27§	31.37 ± 6.08§	52.38 ± 7.49§
Algorithm 2	9.72 ± 1.43§	12.19 ± 1.62§	14.40 ± 2.24§	19.78 ± 4.33§	26.50 ± 2.57§	48.03 ± 9.23§
Algorithm 3	8.95 ± 1.21§	11.22 ± 0.95§	13.93 ± 0.79§	18.24 ± 3.67§	24.74 ± 6.24§	43.44 ± 6.22§
Algorithm 4	10.21 ± 3.95§	12.52 ± 4.01§	16.84 ± 5.60§	24.35 ± 6.85§	32.36 ± 10.04§	50.77 ± 8.31§
Algorithm 5	7.72 ± 1.47≈	8.95 ± 1.55≈	12.12 ± 0.75§	15.55 ± 2.81≈	21.43 ± 2.57≈	40.43 ± 6.82≈
Algorithm 6	7.49 ± 1.38	9.39 ± 1.42	11.78 ± 0.54	14.56 ± 1.87	20.82 ± 2.68	42.72 ± 8.69

Note: "†" and "§" indicate a 0.05 level of significance by Wilcoxon rank sum test. "†", "§" and "≈" denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of Algorithm 6, respectively.

Table 3-7. Average results [± standard deviation (std)] for SSM and the algorithms for synthetic stochastic BA graphs with different error rates in terms of ESR.

Methods \ CI	0.7	0.75	0.8	0.85	0.9	0.95
SSM	24.66 ± 3.39§	30.16 ± 2.28§	37.19 ± 2.82§	46.65 ± 3.55§	60.59 ± 4.65§	85.62 ± 6.59§

Algorithm 1	$9.27 \pm 1.74^{\S}$	$12.83 \pm 1.70^{\S}$	$15.04 \pm 1.78^{\S}$	$19.19 \pm 1.94^{\S}$	$27.18 \pm 2.57^{\S}$	$43.45 \pm 4.39^{\S}$
Algorithm 2	$9.77 \pm 1.19^{\S}$	$12.48 \pm 1.36^{\S}$	$15.55 \pm 2.24^{\S}$	$19.56 \pm 2.15^{\S}$	$24.98 \pm 2.53^{\S}$	$42.62 \pm 5.86^{\S}$
Algorithm 3	$8.92 \pm 1.32^{\S}$	$11.98 \pm 2.09^{\S}$	$15.36 \pm 4.43^{\S}$	$21.19 \pm 7.70^{\S}$	$23.50 \pm 2.23^{\S}$	$38.61 \pm 2.81^{\S}$
Algorithm 4	$10.09 \pm 0.49^{\S}$	$11.91 \pm 1.19^{\S}$	$14.30 \pm 1.49^{\S}$	$17.75 \pm 1.94^{\S}$	$24.95 \pm 3.56^{\S}$	$40.59 \pm 6.18^{\S}$
Algorithm 5	$7.51 \pm 1.53^{\approx}$	$8.75 \pm 1.13^{\approx}$	$11.46 \pm 1.87^{\S}$	$16.04 \pm 1.25^{\S}$	$21.72 \pm 3.28^{\approx}$	$35.22 \pm 6.89^{\S}$
Algorithm 6	7.29 ± 1.52	8.31 ± 1.81	10.51 ± 1.84	14.88 ± 1.63	21.30 ± 5.76	30.86 ± 3.03

Note: "†" and "§" indicate a 0.05 level of significance by Wilcoxon rank sum test. "†", "§" and "≈" denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of Algorithm 6, respectively.

Table 3-8. Average results [± standard deviation (std)] for SSM and the algorithms for synthetic stochastic WS graphs with different error rates in terms of ESR.

Methods \ CI	0.7	0.75	0.8	0.85	0.9	0.95
SSM	$25.58 \pm 0.80^{\S}$	$31.29 \pm 1.10^{\S}$	$38.58 \pm 1.36^{\S}$	$48.42 \pm 1.73^{\S}$	$62.90 \pm 2.24^{\S}$	$88.89 \pm 3.19^{\S}$
Algorithm 1	$10.92 \pm 1.42^{\S}$	$13.43 \pm 2.17^{\S}$	$15.87 \pm 2.68^{\S}$	$18.84 \pm 2.15^{\S}$	$28.85 \pm 6.78^{\S}$	$47.61 \pm 8.80^{\S}$
Algorithm 2	$9.65 \pm 1.08^{\S}$	$12.21 \pm 2.63^{\S}$	$16.25 \pm 3.01^{\S}$	$18.72 \pm 3.01^{\S}$	$26.92 \pm 5.63^{\S}$	$43.57 \pm 9.17^{\S}$
Algorithm 3	$9.89 \pm 1.45^{\S}$	$11.21 \pm 1.20^{\S}$	$15.62 \pm 4.47^{\S}$	$19.34 \pm 5.45^{\S}$	$25.07 \pm 4.68^{\S}$	$45.07 \pm 6.14^{\S}$
Algorithm 4	$9.17 \pm 0.73^{\S}$	$10.95 \pm 1.19^{\S}$	$14.82 \pm 1.52^{\S}$	$17.82 \pm 2.48^{\S}$	$26.04 \pm 4.99^{\S}$	$45.27 \pm 6.83^{\S}$
Algorithm 5	$8.73 \pm 1.10^{\S}$	$9.88 \pm 1.35^{\approx}$	$12.54 \pm 1.02^{\S}$	$16.17 \pm 1.62^{\approx}$	$21.28 \pm 3.46^{\approx}$	$41.83 \pm 4.62^{\S}$
Algorithm 6	8.04 ± 1.21	9.17 ± 1.39	11.22 ± 1.08	15.00 ± 1.62	19.54 ± 2.94	38.86 ± 4.22

Note: "†" and "§" indicate a 0.05 level of significance by Wilcoxon rank sum test. "†", "§" and "≈" denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of Algorithm 6, respectively.

Moreover, a multi-comparison statistical tests using Friedman and Iman–Davenport with 95% confidence level (α=0.05) among the algorithms are also conducted (García et al. 2009) for all synthetic stochastic graphs. As a statistical analysis, Friedman's test is first applied to obtain rankings. To obtain the adjusted *p*-values for each comparison between the control algorithm (the best-performing one) and the other algorithms, Nemenyi, Holm and Shaffer tests are conducted as post-hoc methods (if significant differences are detected). The rankings obtained by Friedman's test are presented in Table 3-9. The *p*-value computed by Friedman's test is 4.64E-11, which is below the significance interval of 95% (α=0.05). Thus, a significant difference exists among the observed results. Post-hoc methods (Nemenyi, Holm and Shaffer tests) are also performed to obtain the adjusted *p*-values. Table 3-10 shows the adjusted *p*-values of the Nemenyi, Holm and Shaffer tests. According to the results of statistical significance in Table 3-9, one can conclude that Algorithm 6 outperforms the other algorithms in terms of average number of samples taken from each edge.

Table 3-9. Average ranking of Friedman's test of comparison algorithms on the stochastic test networks.

Test results	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4	Algorithm 5	Algorithm 6
Average ranking	5.38	4.61	3.77	4.22	1.88	1.11
Ranking	6	5	3	4	2	1

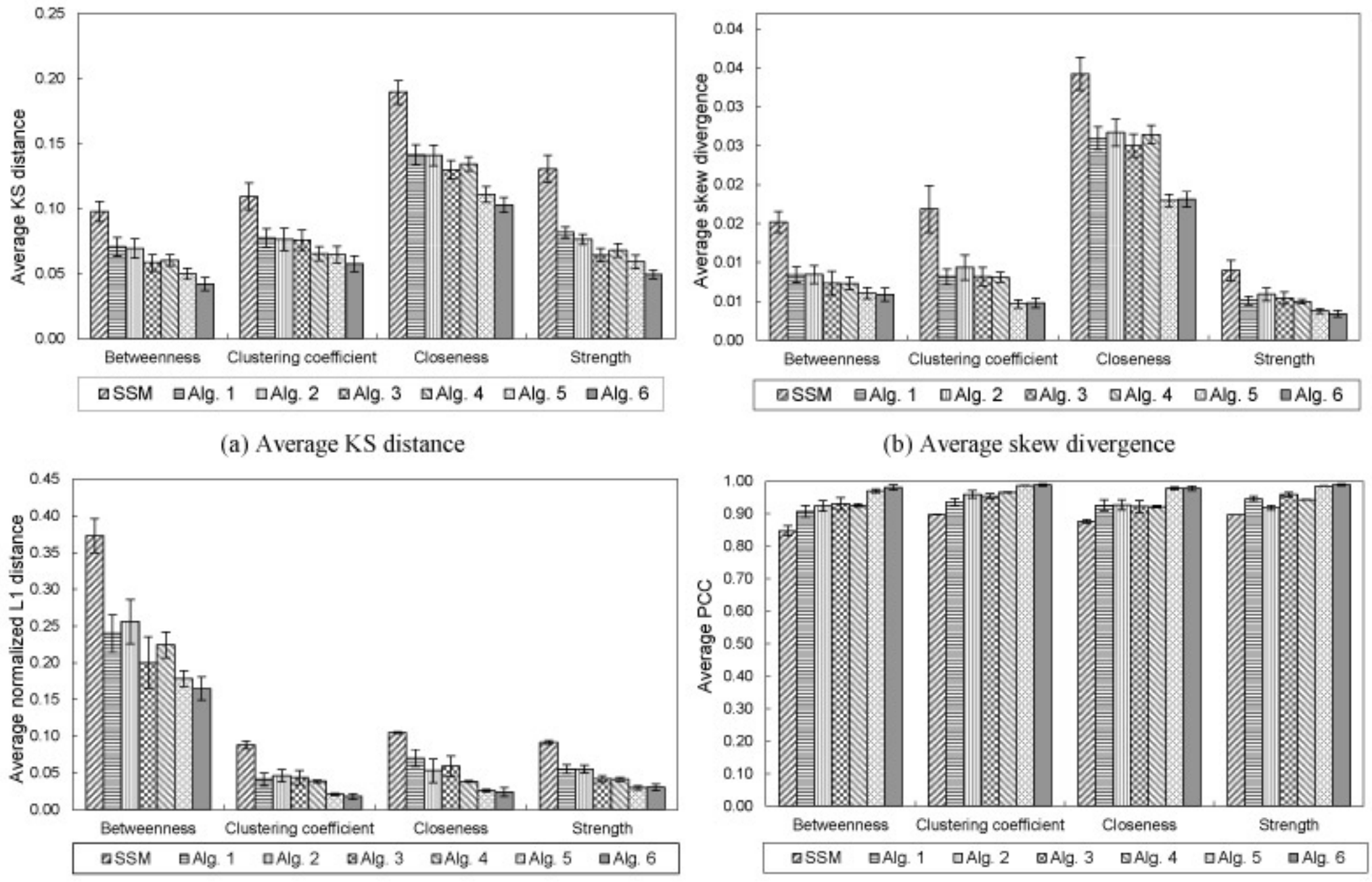
Table 3-10. *p*-values of the comparing algorithms on the stochastic test networks.

i	Hypothesis	Unadjusted *p*-value	*Nemenyi p*-value	*Holm p*-value	*Shaffer p*-value
15	Algorithm 1 vs. Algorithm 6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
14	Algorithm 1 vs. Algorithm 5	0.00E+00	0.00E+00	0.00E+00	0.00E+00
13	Algorithm 2 vs. Algorithm 6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
12	Algorithm 4 vs. Algorithm 6	1.00E-06	9.00E-06	7.00E-06	6.00E-06
11	Algorithm 2 vs. Algorithm 5	1.30E-05	1.90E-04	1.40E-04	1.27E-04
10	Algorithm 3 vs. Algorithm 6	1.90E-05	2.85E-04	1.90E-04	1.90E-04
9	Algorithm 4 vs. Algorithm 5	1.83E-04	2.74E-03	1.65E-03	1.28E-03
8	Algorithm 3 vs. Algorithm 5	2.45E-03	3.68E-02	1.96E-02	1.72E-02
7	Algorithm 1 vs. Algorithm 3	9.78E-03	1.47E-01	6.85E-02	6.85E-02
6	Algorithm 1 vs. Algorithm 4	6.14E-02	9.21E-01	3.68E-01	3.68E-01
5	Algorithm 2 vs. Algorithm 3	1.81E-01	2.72E+00	9.07E-01	7.26E-01
4	Algorithm 5 vs. Algorithm 6	2.12E-01	3.18E+00	9.07E-01	8.49E-01
3	Algorithm 1 vs. Algorithm 2	2.12E-01	3.18E+00	9.07E-01	8.49E-01
2	Algorithm 3 vs. Algorithm 4	4.76E-01	7.14E+00	9.52E-01	9.52E-01
1	Algorithm 2 vs. Algorithm 4	5.33E-01	7.99E+00	9.52E-01	9.52E-01

Note: Nemenyi's procedure rejects those hypotheses that have an unadjusted p-value ≤ 6.67E-03; Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 1.67E-02; Shaffer's procedure rejects those hypotheses that have an unadjusted p-value ≤ 6.67E-03.

3.4.3.3. **Experiment III**

This experiment is conducted to study the performance of the algorithms (Alg. 1, Alg. 2, Alg. 3, Alg. 4, Alg. 5 and Alg. 6) and standard sampling method (SSM) for sampling from stochastic graphs in terms of different distance metrics. For this experimentation, different synthetic stochastic graphs (BA, WS and ER) with size from 1000 to 10000 are used. The results of this experimentation are averages taken over all the test networks with respect to different distance metrics: Kolmogorov-Smirnov (KS) distance, skew divergence (SD), normalized L_1 distance (ND) and Pearson's correlation coefficient (PCC) for estimated measures including: betweenness, clustering coefficient, closeness and strength. The results of average and standard error with 95% confidence interval of mentioned metrics are given in Figure 3-7 through Figure 3-9 for synthetic stochastic graphs. From Figure 3-7 through Figure 3-9, Algorithm 6 outperforms other algorithms and standard sampling method for all test graphs and standard sampling method achieves the low accuracy in terms of KS, SD, ND and PCC for mentioned metrics. For all test graphs, the results for betweenness, clustering coefficient and strength is more reliable than the results for closeness in terms of KS, SD and ND. Also, the results for betweenness in terms of ND are larger than the results for other metrics.



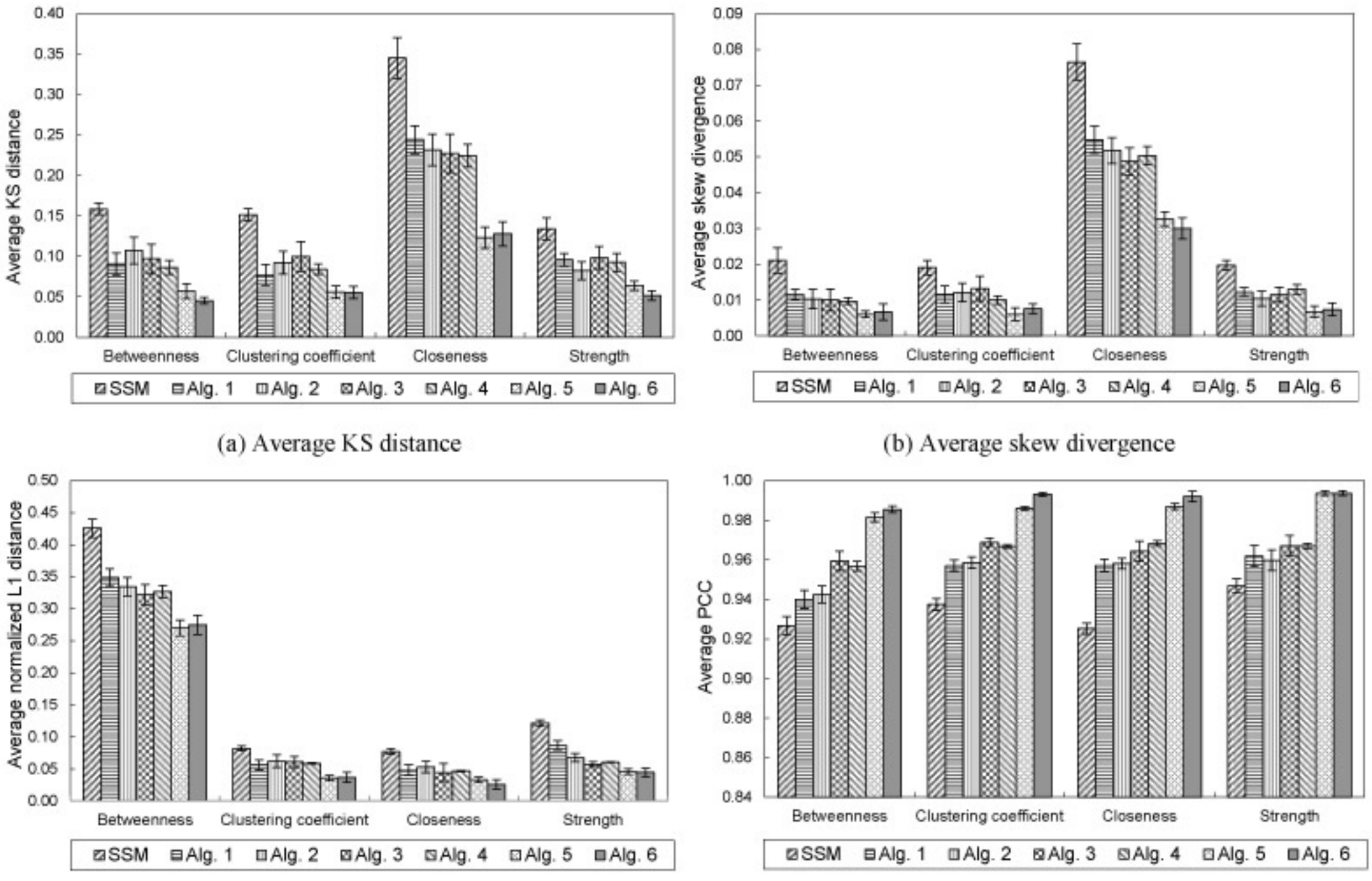
(a) Average KS distance

(b) Average skew divergence

(c) Average normalized L_1 distance

(d) Average Pearson's correlation coefficient

Figure 3-7. Comparing results of average KS distances, average skew divergence, average normalized L_1 distance and average Pearson's correlation coefficient over betweenness, clustering coefficient, closeness and strength for synthetic stochastic ER graphs.



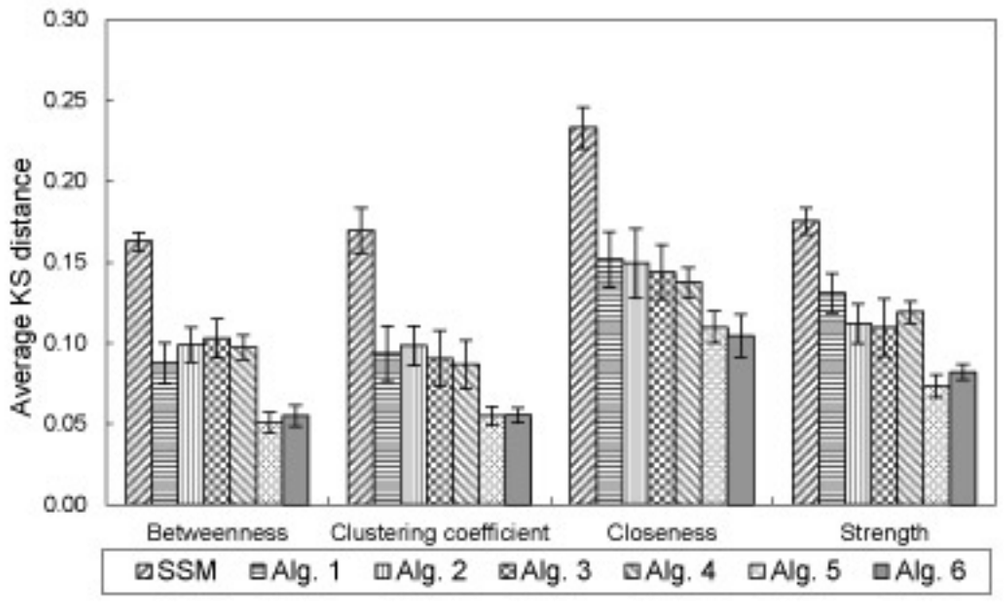
(a) Average KS distance

(b) Average skew divergence

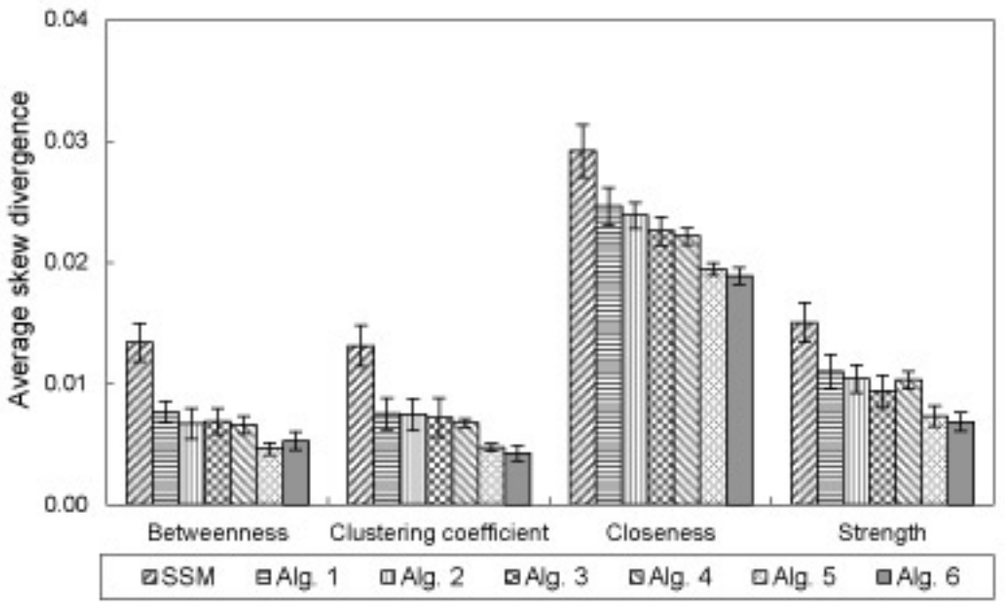
(c) Average normalized L_1 distance

(d) Average Pearson's correlation coefficient

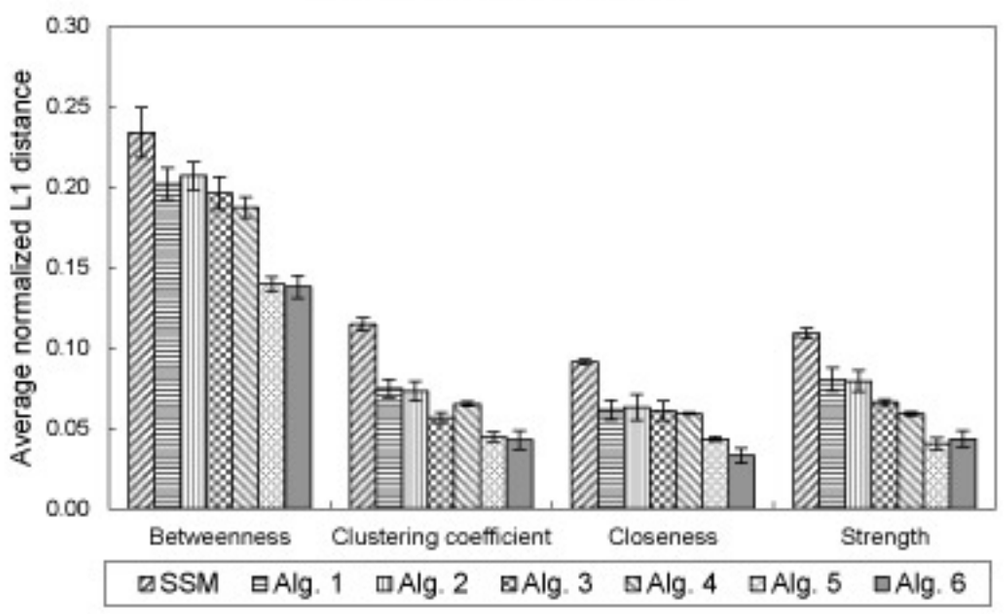
Figure 3-8. Comparing results of average KS distance, average skew divergence, average normalized L_1 distance and average Pearson's correlation coefficient over betweenness, clustering coefficient, closeness and strength for synthetic stochastic BA graphs.



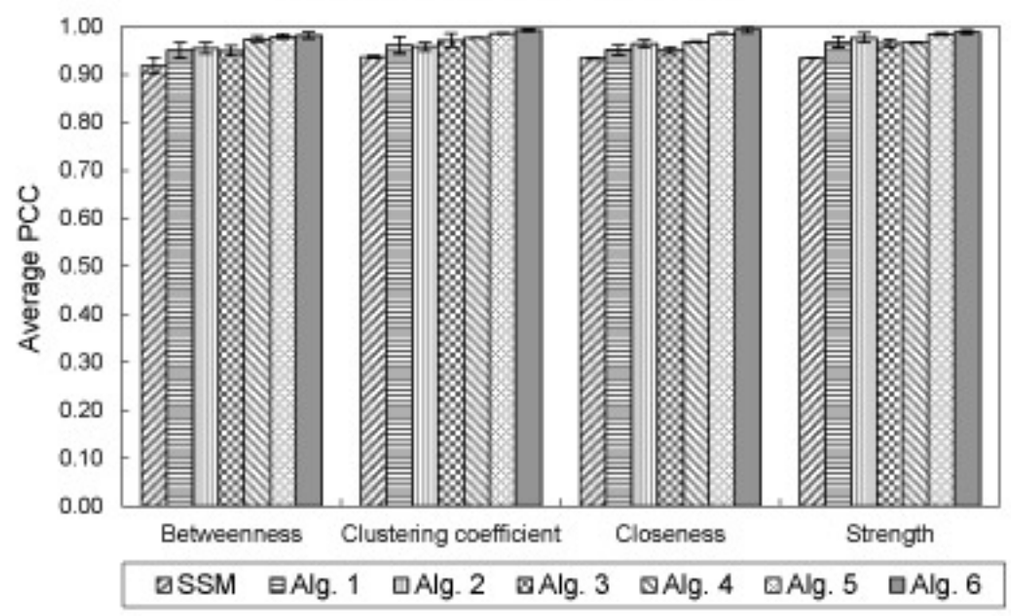
(a) Average KS distance



(b) Average skew divergence



(c) Average normalized L_1 distance



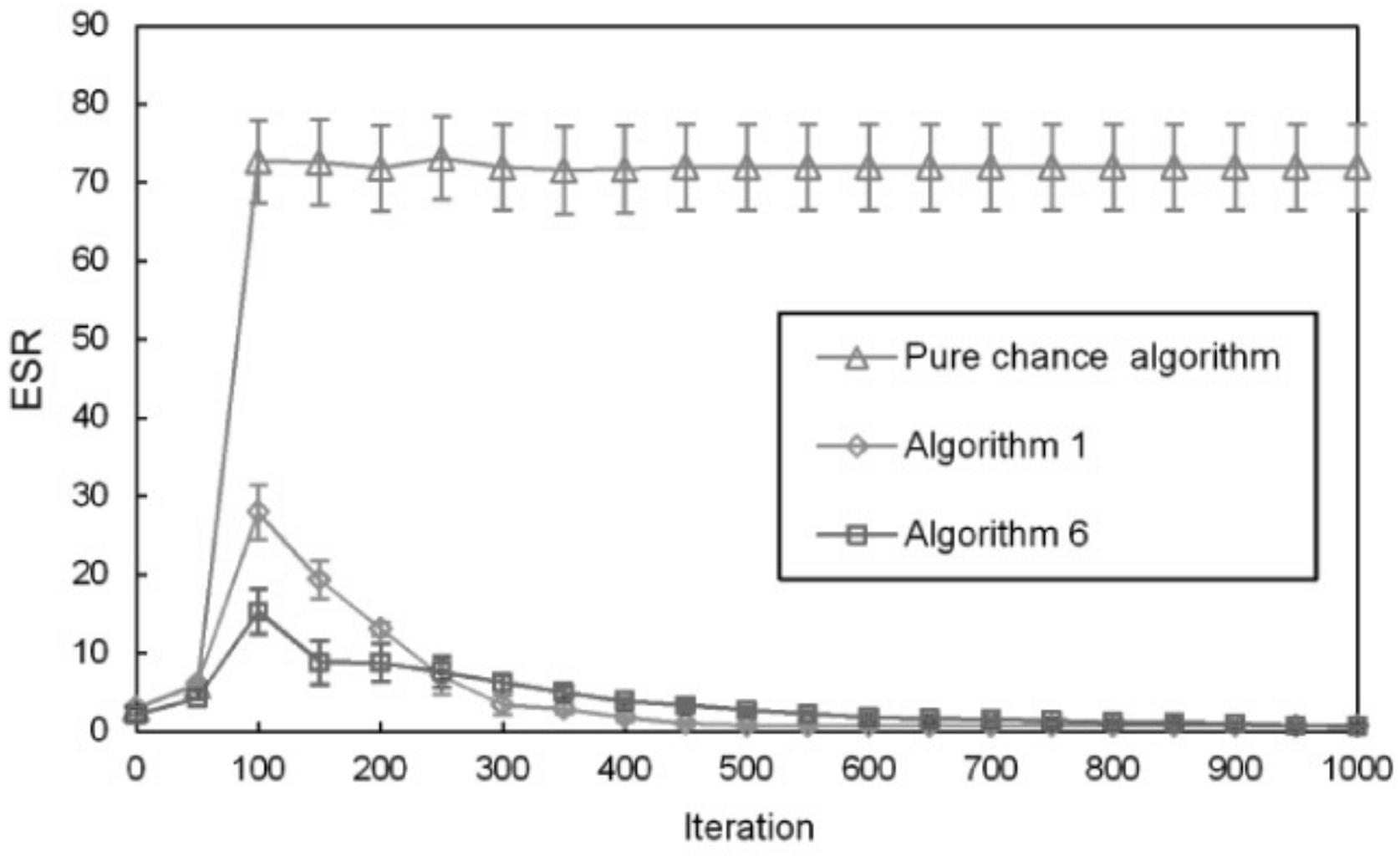
(d) Average Pearson's correlation coefficient

Figure 3-9. Comparing results of average KS distance, average skew divergence, average normalized L_1 distance and average Pearson's correlation coefficient over betweenness, clustering coefficient, closeness and strength for synthetic stochastic WS graphs.

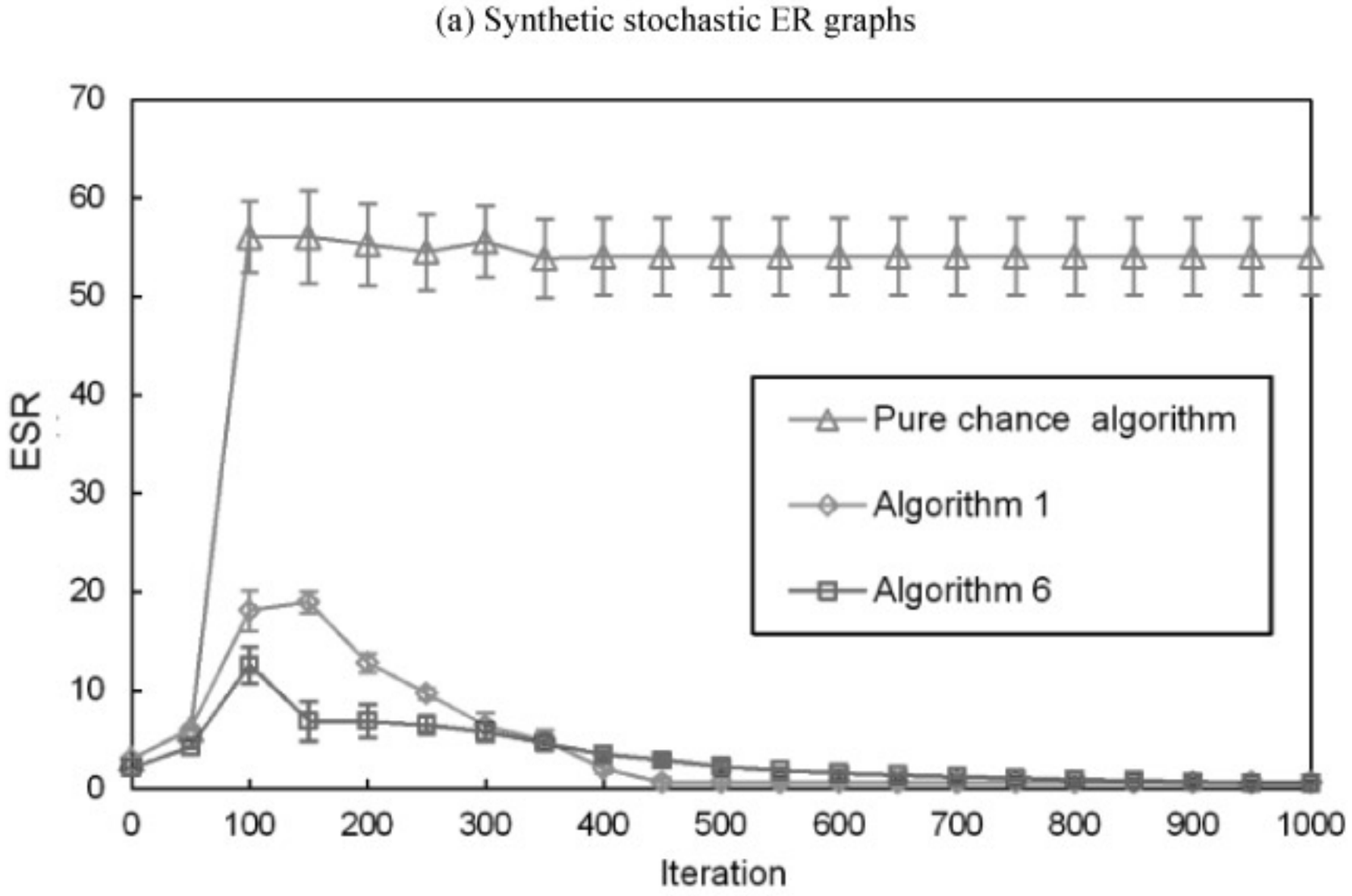
3.4.3.4. **Experiment IV**

This experiment is conducted to compare the algorithms (Algorithm 1 and algorithm 6) with the pure chance algorithm (algorithm 1 in which the leaning automaton residing in each node is replaced by a pure chance automaton) with respect to the number of samples taken from the edges of the graph. In pure chance automaton, the actions are chosen with equal probabilities. In order to perform this experiment, we plot average number of samples taking from each edge of graph (ESR) during the execution of the algorithms for computing network measures versus iteration number for both algorithms (Algorithm 1 and Algorithm 6) and pure chance algorithm. The results of this experiment are taken averages for each synthetic stochastic test graphs and are given in Figure 3-10. In Figure 3-10, the points along the curves show the average value and the error bars represent 95% confidence interval. As it is shown, as the algorithms proceeds, the number of samples taken from each edge (ESR) abundantly increases in initial iterations for all algorithms and then reduces and gradually approaches zero for Algorithm 1 and Algorithm 6, however ESR value for pure chance algorithm remains unchanged. This implies that for the algorithms, the average number of samples taken from each edge of the graph gradually approaches zero. And also it is done with fewer numbers of samples taken from the graph as compared to the case where learning is absent that indicates the important impact of learning on the superiority of the learning automaton based algorithms for computing network measures in stochastic

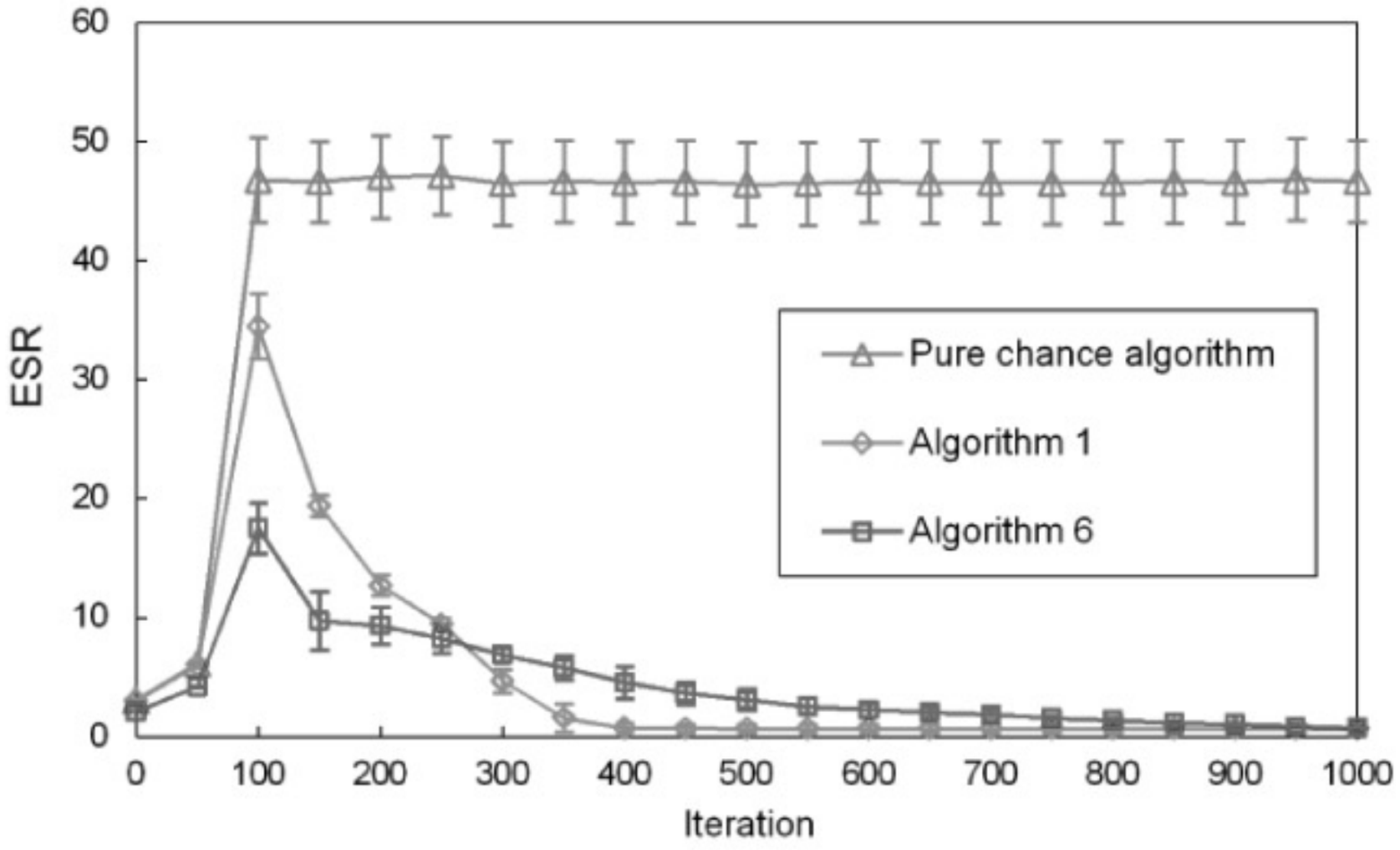
graphs. From the error bars shown in Figure 3-10, we conclude that the variances in the results decreases as the iteration number increases. According to the results for comparison among network models, we can see that for Synthetic stochastic ER graphs, the average number of samples taking from each edge of graph is much more than that of both Synthetic stochastic BA and WS graphs. Similar result can be obtained for other algorithms.



(a) Synthetic stochastic ER graphs



(b) Synthetic stochastic BA graphs

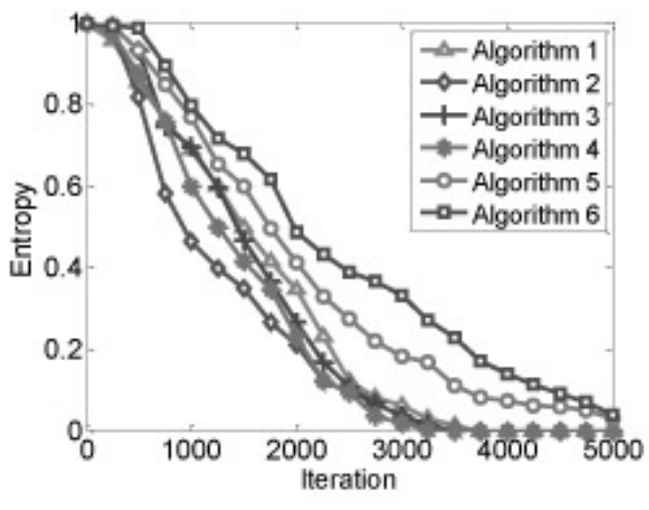


(c) Synthetic stochastic WS graphs

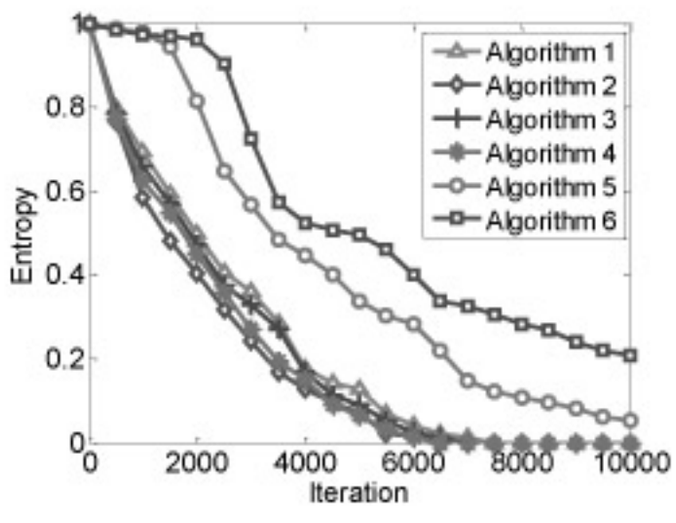
Figure 3-10. The plot of average cumulative *ESR* versus iteration number for stochastic test graphs.

3.4.3.5. **Experiment V**

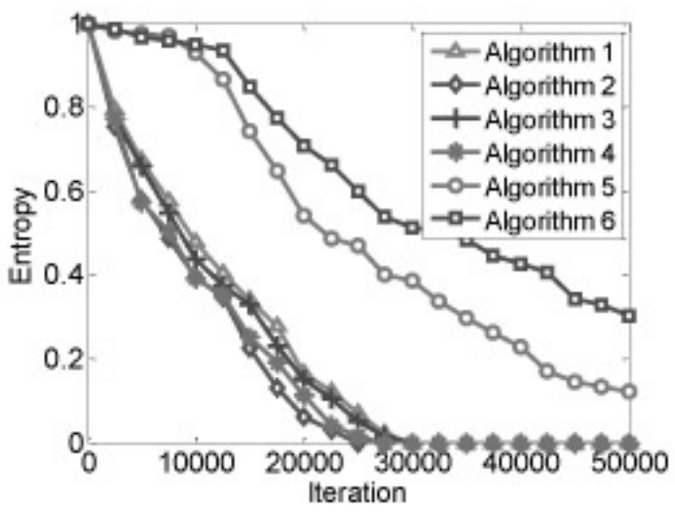
In this experiment, we study the convergence behavior of the algorithms using information entropy (as defined in equation (3-5)) of probability vector of learning automata residing in the nodes of the graph responsible for deciding whether or not taking samples from the edges of graph. For this purpose, for each algorithm we plot the average information entropy taken over all leaning automata residents in the nodes versus iteration number as given in Figure 3-11 for three kinds of synthetic stochastic graphs as mentioned before (ER-2000, ER-5000, ER-10000; BA-2000, BA-5000, BA-10000; WS-2000, WS-5000, WS-10000). As it is shown, for every algorithm the average entropy decreases as the algorithm proceeds. Using the fact the average entropy decreases as the algorithm proceeds and also the result of previous experiment, we may conclude that the learning automaton of each node gradually converges to the action of "do not take sample from the edges of chosen node". We can also conclude that Algorithm 2 has the highest and Algorithm 6 has the lowest speed of convergence. The convergence behaviors of the algorithms among the network models indicate that the difference between algorithms 5-6 and algorithms 1-4 for synthetic stochastic small world graphs is more obvious than other network models.



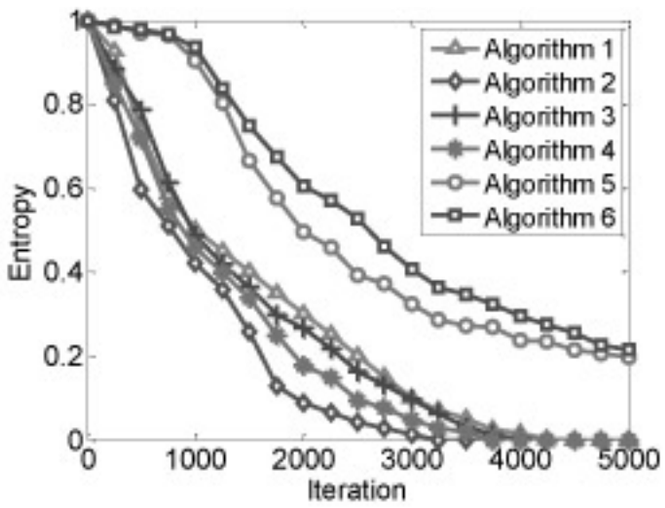
(a) ER-2000



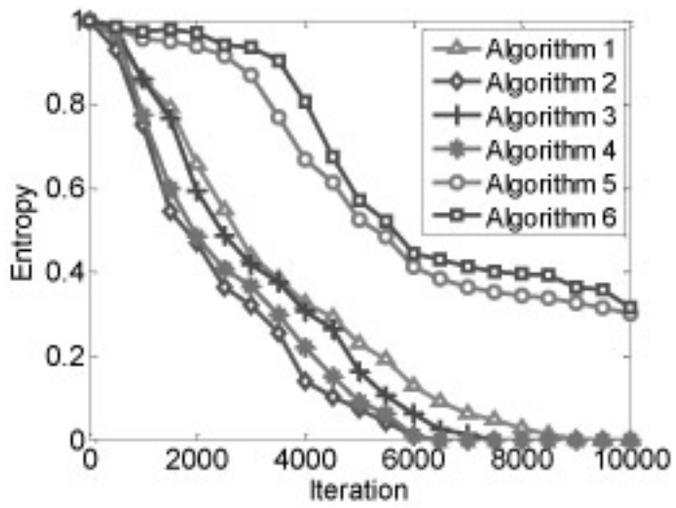
(b) ER-5000



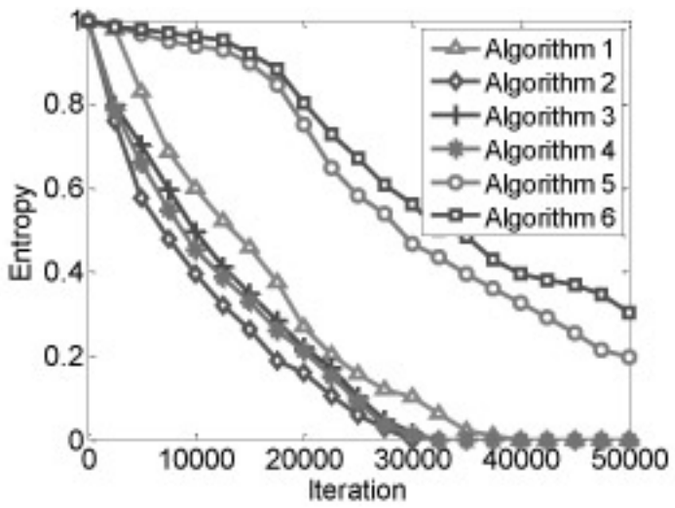
(c) ER-10000



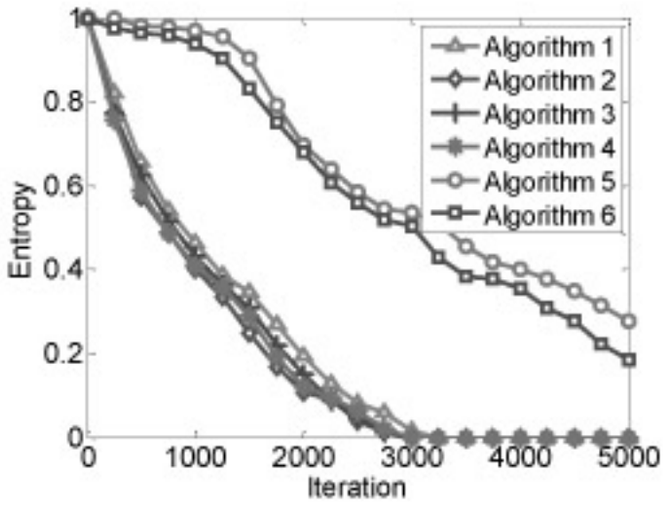
(d) BA-2000



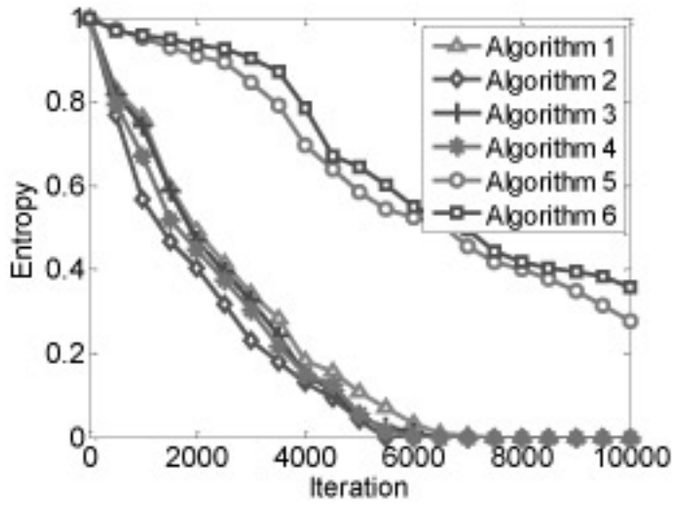
(e) BA-5000



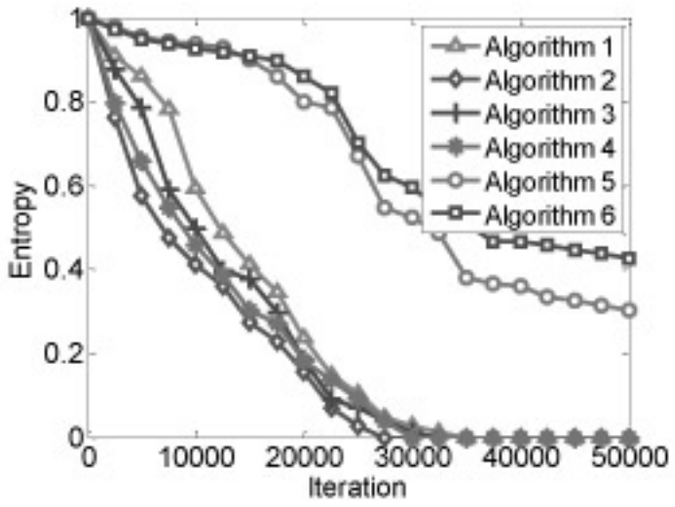
(f) BA-10000



(g) WS-2000



(h) WS-5000



(i) WS-10000

Figure 3-11. Comparison of convergence behaviors of the algorithms with respect to the information entropy for synthetic stochastic graphs.

3.4.4. **Discussion**

This study argued about the important role of activities of online users as the main features of online social networks for generating and sharing information through social networks. In general, activities of users in online social networks such as interacting with their friends, answering comments of their friends, participating in online communities, posting and sharing new information contents on the page of their friends, visiting profile pages of their friends, taking comment or like on a post of their friends are considered as the weights associated with the edges of graph and due to nature of user activities in social network these weights may be uncertain, unpredictable and time-varying. Therefore in this study, we that stochastic graphs, in which weights associated with the edges are random variables, may be a better candidate as a graph model for social network analysis. We assumed that the structural and behavioral parameters of online social networks which are time varying parameters can be observed by the network analyzers. Using the observed parameters, the network analyzers can compute different measurements such as strength, closeness and betweenness in an ongoing fashion. For example, one may want to find a user which can influence a large number of users by spreading of influence in a social network for marketing goals or control the information flows over communication between other users for managing misinformation. This may be easily realized by computing network centralities using one of the algorithms as the network operates. The algorithms for computing network measures can be used by the network analyzers as a means for observing the time varying parameters of the network for the purpose of network's measurements computations.

In the simulation, it was showed that modeling social networks as stochastic graph models by applying the framework of learning automation can be very beneficial to reach good results with a certain confidence level. Since, in many fundamental studies, researchers generate synthetic graphs similar to real networks in order to further simulations and investigate a number of phenomena on a computer generated graph models in order to be able to exploit their structures and dynamics of a network in a systematic manner, in the simulation, it was used several synthetic stochastic networks such as *Barabási-Albert* model (Barabási and Albert 1999) as a synthetic scale-free network, *Watts–Strogatz* model (Watts and Strogatz 1998) as a synthetic small world network and *Erdös–Rényi* model (Erdos and Rényi 1960) as a synthetic random network in which edge weights are assumed to be random variables with exponential distributions. Furthermore, it was indicated that how the algorithms can collect information from the network in order to find good estimates for the network's measurements using fewer numbers of samples than that of standard sampling methods. The overall results confirm that using two set of learning automata can be achieve better results than that of one set of learning automata in terms of accuracy and convergence behavior.

Analyzing social networks and its user behavior with stochastic graph models can be applied for some applications. Such application can be in viral marketing in such a way that companies want to exploit user behavior models and then spread or promote their new products or services to predict the user interactions and process of adoption of an idea or spread them. Another example can be in domain of knowledge based systems by understanding the patterns of user participation to generate new contents, propagating information on the networks via users, detecting active users from spamming users, attracting new users and keeping some users, predicting the trends of topics in user communities, and performing efficient content management.

In summary, due to important role of modeling user behavior of online social networks in quantitative manner and variety of user interactions on the structure of social networks, analyzing social networks with stochastic graph models may be a better candidate as a graph model for analysis of social network and its user behaviors which can take into consideration the continuum of behavioral parameters of network occurring over time.

3.5. **Conclusion and future directions**

3.5.1. **Conclusion**

The conventional social network models solely consider either the existence of connections between users in the form of binary networks or consider fixed weights for the edges in the form of weighted networks. In this chapter, stochastic graph has been introduced as a model for social networks and then redefined some of the social network measurements to be applicable to stochastic graphs. Also several algorithms based on learning automata were designed for finding good estimations of these measurements for the purpose of analysis. We believe that the approach introduced in this chapter for modeling and analysis of social

networks can provide a better way for studying online social networks such analysis of user behavior and online human activities.

3.5.2. **Future directions**

This book has made a positive contribution towards using stochastic graph as a graph model of social networks. Besides, as the notable achievements of this book, it opens up several avenues and directions for further study and research, some of which are given below.

- In this book, we defined several network measures for stochastic graph and then introduced several algorithms for computing these measures. Closed form for the distribution of network measures in stochastic graphs were obtained when the weights associated with the edge weights are exponential random variables. Mathematical analysis of these network measures can be pursued, considering other type of distributions such as Pareto distribution, Weibull distribution, and Rayleigh distribution.
- In this book, we introduced several learning automata based algorithms for computing estimations of the networks measures in stochastic graphs. These algorithms can be extended, considering other type of learning automata models such as continuous action set learning automata, Bayesian learning automata, and so on.
- In this book, we assumed that the structural and behavioral parameters of online social networks are time varying parameters which can be observed by the network. There are situations that the value of these parameters may be noisy because of existence of fake/spam users or may not be available for some reasons such as privacy setting limitation of some social networks. The algorithms presented in this book need to modify to take into consideration such situations.

BIBLIOGRAPHY

Abramowitz, Milton, and Irene A. Stegun. 1974. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Dover, New York: Courier Corporation.

Ahmed, Nesreen K., Jennifer Neville, and Ramana Kompella. 2014. "Network Sampling: From Static to Streaming Graphs." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8 (2): 7.

Aliakbary, Sadegh, Jafar Habibi, and Ali Movaghar. 2015. "Feature Extraction from Degree Distribution for Comparison and Analysis of Complex Networks." *The Computer Journal* 58 (9): 2079–91.

Amiri, F., N. Yazdani, H. Faili, and A. Rezvanian. 2013. "A Novel Community Detection Algorithm for Privacy Preservation in Social Networks." In *Intelligent Informatics*, edited by A. Abraham, 18:443–450. Advances in Intelligent Systems and Computing.

B Alt, F. 1982. "Bonferroni Inequalities and Intervals." *Encyclopedia of Statistical Sciences*.

Badie, Reza, Abolfazl Aleahmad, Masoud Asadpour, and Maseud Rahgozar. 2013. "An Efficient Agent-Based Algorithm for Overlapping Community Detection Using Nodes' Closeness." *Physica A: Statistical Mechanics and Its Applications* 392 (20): 5231–5247.

Barabási, A. L., and R. Albert. 1999. "Emergence of Scaling in Random Networks." *Science* 286 (5439): 509–512.

Barto, A. G., R. S. Sutton, and C. W. Anderson. 1983. "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems." *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13 (5): 834–46.

Beigy, H., and M. R. Meybodi. 2006. "Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problems." *International Journal of Uncertainty Fuzziness And Knowledge Based Systems* 14 (5): 591–615.

Bild, David R., Yue Liu, Robert P. Dick, Z. Morley Mao, and Dan S. Wallach. 2015. "Aggregate Characterization of User Behavior in Twitter and Analysis of the Retweet Graph." *ACM Transactions on Internet Technology (TOIT)* 15 (1): 4.

Bonferroni, Carlo E. 1936. *Teoria Statistica Delle Classi E Calcolo Delle Probabilita*. Libreria internazionale Seeber.

Borgatti, Stephen P. 2005. "Centrality and Network Flow." *Social Networks* 27 (1): 55–71.

Buccafurri, Francesco, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. 2015. "Comparing Twitter and Facebook User Behavior: Privacy and Other Aspects." *Computers in Human Behavior* 52: 87–95.

Cha, Meeyoung, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. 2007. "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System." In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 1–14. ACM.

Costa, L.F., F.A. Rodrigues, G. Travieso, and P.R.V. Boas. 2007. "Characterization of Complex Networks: A Survey of Measurements." *Advances in Physics* 56 (1): 167–242.

Damerchilu, B., M. S. Norouzzadeh, and M. R. Meybodi. 2016. "Motion Estimation Using Learning Automata." *Machine Vision and Applications*, 1–15.

Daniel, Wayne W. 2000. *Applied Nonparametric Statistics*. 2nd ed. Duxbury Thomson Learning.

Ding, Fei, Hui Cheng, Xia-Meng Si, Yun Liu, Fei Xiong, and Bo Shen. 2010. "Read and Reply Behaviors in a BBS Social Network." In *2010 2nd International Conference on Advanced Computer Control (ICACC)*, 4:571–576. IEEE.

Erdos, P., and A. Rényi. 1960. "On the Evolution of Random Graphs." *Publications of the Mathematical Institute of the Hungarian Academy of Sciences* 5: 17–61.

Falck-Ytter, Martin, and Harald Øverby. 2012. "An Empirical Study of Valuation and User Behavior in Social Networking Services." In *World Telecommunications Congress (WTC)*, 1–6. IEEE.

Feng, Ziming, Feng Cong, Kailong Chen, and Yong Yu. 2013. "An Empirical Study of User Behaviors on Pinterest Social Network." In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 1:402–409. IEEE.

Florescu, Ionut, and Ciprian A. Tudor. 2013. *Handbook of Probability*. John Wiley & Sons.

Fortunato, S. 2010. "Community Detection in Graphs." *Physics Reports* 486 (3–5): 75–174.

Freeman, L. C. 1979. "Centrality in Social Networks Conceptual Clarification." *Social Networks* 1 (3): 215–239.

Galuba, Wojciech, Karl Aberer, Dipanjan Chakraborty, Zoran Despotovic, and Wolfgang Kellerer. 2010. "Outtweeting the Twitterers-Predicting Information Cascades in Microblogs." In *Proceedings of the 3rd Conference on Online Social Networks*, 39:1–9.

García, S., D. Molina, M. Lozano, and F. Herrera. 2009. "A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization." *Journal of Heuristics* 15 (6): 617–644.

Girvan, M., and M.E.J. Newman. 2002. "Community Structure in Social and Biological Networks." *Proceedings of the National Academy of Sciences* 99 (12): 7821–26.

Golder, Scott A., Dennis M. Wilkinson, and Bernardo A. Huberman. 2007. "Rhythms of Social Interaction: Messaging within a Massive Online Network." In *Communities and Technologies 2007*, 41–66. Springer.

Guo, Lei, Enhua Tan, Songqing Chen, Xiaodong Zhang, and Yihong Eric Zhao. 2009. "Analyzing Patterns of User Content Generation in Online Social Networks." In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 369–378. ACM.

Gyarmati, László, and Tuan Anh Trinh. 2010. "Measuring User Behavior in Online Social Networks." *IEEE Network* 24 (5): 26–31.

Hasanzadeh, M, and Mohammad Reza Meybodi. 2014. "Grid Resource Discovery Based on Distributed Learning Automata." *Computing* 96 (9): 909–22.

Hasanzadeh, M, Mohammad Reza Meybodi, and Mohammad Mehdi Ebadzadeh. 2013. "Adaptive Cooperative Particle Swarm Optimizer." *Applied Intelligence* 39 (2): 397–420.

Hasanzadeh, Mohammad, and Mohammad Reza Meybodi. 2015. "Distributed Optimization Grid Resource Discovery." *The Journal of Supercomputing* 71 (1): 87–120.

Hochberg, Yosef. 1988. "A Sharper Bonferroni Procedure for Multiple Tests of Significance." *Biometrika* 75 (4): 800–802.

Holm, Sture. 1979. "A Simple Sequentially Rejective Multiple Test Procedure." *Scandinavian Journal of Statistics*, 65–70.

Iman, Ronald L., and James M. Davenport. 1980. "Approximations of the Critical Region of the Fbietkan Statistic." *Communications in Statistics-Theory and Methods* 9 (6): 571–595.

Janssen, Jeannette, Matt Hurshman, and Nauzer Kalyaniwalla. 2012. "Model Selection for Social Networks Using Graphlets." *Internet Mathematics* 8 (4): 338–363.

Jin, Long, Yang Chen, Tianyi Wang, Pan Hui, and Athanasios V. Vasilakos. 2013. "Understanding User Behavior in Online Social Networks: A Survey." *IEEE Communications Magazine* 51 (9): 144–50.

Johnson, Norman L., Samuel Kotz, and N. Balakrishnan. 1994. "Continuous Univariate Distributions , Vol. 1John Wiley & Sons." *New York*, 163.

Kempe, D., J. Kleinberg, and É Tardos. 2003. "Maximizing the Spread of Influence through a Social Network." In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137–46.

Khomami, M. M. D., A. Rezvanian, and M. R. Meybodi. 2016. "Distributed Learning Automata-Based Algorithm for Community Detection in Complex Networks." *International Journal of Modern Physics B* 30: 1650042.

Krishna, P. Venkata, Sudip Misra, Dheeraj Joshi, Anant Gupta, and Mohammad S. Obaidat. 2014. "Secure Socket Layer Certificate Verification: A Learning Automata Approach." *Security and Communication Networks* 17 (11): 1712–18.

Kruskal, William H., and W. Allen Wallis. 1952. "Use of Ranks in One-Criterion Variance Analysis." *Journal of the American Statistical Association* 47 (260): 583–621.

Kumar, N., and Jong-Hyouk Lee. 2015. "Collaborative-Learning-Automata-Based Channel Assignment With Topology Preservation for Wireless Mesh Networks Under QoS Constraints." *IEEE Systems Journal* 9 (3): 675–85.

Kumar, N., S. Misra, M. Obaidat, J. Rodrigues, and B. Pati. 2014. "Networks of Learning Automata for the Vehicular Environment: A Performance Analysis Study." *IEEE Wireless Communications* 21 (6): 41–47.

Kumar, N., S. Misra, and M.S. Obaidat. 2015. "Collaborative Learning Automata-Based Routing for Rescue Operations in Dense Urban Regions Using Vehicular Sensor Networks." *IEEE Systems Journal* 9 (3): 1081–90.

Kumar, Narendra, Joun-Ho Lee, and Joel JPC Rodrigues. 2015. "Intelligent Mobile Video Surveillance System as a Bayesian Coalition Game in Vehicular Sensor Networks: Learning Automata Approach." *IEEE Transactions on Intelligent Transportation Systems* 16 (3): 1148–61.

Kwak, Haewoon, Changhyun Lee, Hosung Park, and Sue Moon. 2010. "What Is Twitter, a Social Network or a News Media?" In *Proceedings of the 19th International Conference on World Wide Web*, 591–600. ACM.

Lakshmivarahan, S., and M. A. L. Thathachar. 1976. "Bounds on the Convergence Probabilities of Learning Automata." *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 6 (11): 756–763.

Leskovec, Jure, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. 2008. "Microscopic Evolution of Social Networks." In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 462–470. ACM.

Leskovec, Jure, Mary McGlohon, Christos Faloutsos, Natalie S. Glance, and Matthew Hurst. 2007. "Patterns of Cascading Behavior in Large Blog Graphs." In *SDM*, 7:551–56. SIAM.

Li, Yongli, Chong Wu, Xiaoyu Wang, and Peng Luo. 2014. "A Network-Based and Multi-Parameter Model for Finding Influential Authors." *Journal of Informetrics* 8 (3): 791–799.

Liu, Han, Atif Nazir, Jinoo Joung, and Chen-Nee Chuah. 2013. "Modeling/Predicting the Evolution Trend of Osn-Based Applications." In *Proceedings of the 22nd International Conference on World Wide Web*, 771–780. ACM.

Liu, Z., C. Jiang, J. Wang, and H. Yu. 2015. "The Node Importance in Actual Complex Networks Based on a Multi-Attribute Ranking Method." *Knowledge-Based Systems* 84: 56–66.

Luo, Peng, Yongli Li, Chong Wu, and Guijie Zhang. 2015. "Toward Cost-Efficient Sampling Methods." *International Journal of Modern Physics C* 26 (5): 1550050.

Mahdaviani, M., J.K. Kordestani, A. Rezvanian, and M.R. Meybodi. 2015. "LADE: Learning Automata Based Differential Evolution." *International Journal on Artificial Intelligence Tools* 24 (6): 1550023.

Mirsaleh, Mehdi Rezapoor, and Mohammad Reza Meybodi. 2016a. "A Michigan Memetic Algorithm for Solving the Community Detection Problem in Complex Network." *Neurocomputing* 214: 535–45.

———. 2016b. "A New Memetic Algorithm Based on Cellular Learning Automata for Solving the Vertex Coloring Problem." *Memetic Computing* 8 (3): 211–22.

Misra, Sudip, Shankha Subhra Chatterjee, and Mohsen Guizani. 2015. "Stochastic Learning Automata-Based Channel Selection in Cognitive Radio/Dynamic Spectrum Access for WiMAX Networks." *International Journal of Communication Systems* 28 (5): 801–17.

Mofrad, Mohammad Hasanzadeh, Sana Sadeghi, Alireza Rezvanian, and Mohammad Reza Meybodi. 2015. "Cellular Edge Detection: Combining Cellular Automata and Cellular Learning Automata." *AEU-International Journal of Electronics and Communications* 69 (9): 1282–1290.

Moradabadi, B., and M. R. Meybodi. 2016. "Link Prediction Based on Temporal Similarity Metrics Using Continuous Action Set Learning Automata." *Physica A: Statistical Mechanics and Its Applications* 460 (15): 361–73.

Moradabadi, Behnaz, and Hamid Beigy. 2014. "A New Real-Coded Bayesian Optimization Algorithm Based on a Team of Learning Automata for Continuous Optimization." *Genetic Programming and Evolvable Machines* 15 (2): 169–193.

Moradabadi, Behnaz, Mohammad Mahdi Ebadzadeh, and Mohammad Reza Meybodi. 2016. "A New Real-Coded Stochastic Bayesian Optimization Algorithm for Continuous Global Optimization." *Genetic Programming and Evolvable Machines* 17 (2): 145–167.

Morales, A. J., J. C. Losada, and R. M. Benito. 2012. "Users Structure and Behavior on an Online Social Network during a Political Protest." *Physica A: Statistical Mechanics and Its Applications* 391 (21): 5244–5253.

Morshedlou, H., and M.R. Meybodi. 2014. "Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments." *IEEE Transactions on Cloud Computing* 2 (2): 156–67.

Mousavian, A., A. Rezvanian, and M. R. Meybodi. 2013. "Solving Minimum Vertex Cover Problem Using Learning Automata." In *13th Iranian Conference on Fuzzy Systems (IFSC 2013)*, 1–5.

Mousavian, Aylin, Alireza Rezvanian, and Mohammad Reza Meybodi. 2014. "Cellular Learning Automata Based Algorithm for Solving Minimum Vertex Cover Problem." In *2014 22nd Iranian Conference on Electrical Engineering (ICEE)*, 996–1000. IEEE.

Narendra, K. S., and M. A. L. Thathachar. 1989. *Learning Automata: An Introduction*. Printice-Hall.

Narendra, Kumpati S., and Mandayam AL Thathachar. 1989. *Learning Automata: An Introduction*. Prentice-Hall.

Nazir, Atif, Saqib Raza, and Chen-Nee Chuah. 2008. "Unveiling Facebook: A Measurement Study of Social Network Based Applications." In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, 43–56. ACM.

Newman, M.E.J. 2005. "A Measure of Betweenness Centrality Based on Random Walks." *Social Networks* 27 (1): 39–54.

Nicopolitidis, P. 2015. "Performance Fairness across Multiple Applications in Wireless Push Systems." *International Journal of Communication Systems* 28 (1): 161–66.

Papoulis, A., and S. Pillai. 2002. *Probability, Random Variables and Stochastic Processes*. 4th ed. McGraw Hill.

Ross, SM. 2009. *Introduction to Probability Models, Tenth Edition*. 10th edition. Amsterdam ; Boston: Academic Press.

Rummery, Gavin A., and Mahesan Niranjan. 1994. *On-Line Q-Learning Using Connectionist Systems*. University of Cambridge, Department of Engineering.

Safavi, S. M., M. R. Meybodi, and M. Esnaashari. 2014. "Learning Automata Based Face-Aware Mobicast." *Wireless Personal Communications* 77 (3): 1923–33.

Saghiri, A. M., and M. R. Meybodi. 2016. "An Approach for Designing Cognitive Engines in Cognitive Peer-to-Peer Networks." *Journal of Network and Computer Applications* 70: 17–40.

Schwartz, Anton. 1993. "A Reinforcement Learning Method for Maximizing Undiscounted Rewards." In *Proceedings of the Tenth International Conference on Machine Learning*, 298:298–305.

Shaffer, Juliet Popper. 1995. "Multiple Hypothesis Testing." *Annual Review of Psychology* 46: 561.

Shen, Jianqiang, Oliver Brdiczka, and Yiye Ruan. 2013. "A Comparison Study of User Behavior on Facebook and Gmail." *Computers in Human Behavior* 29 (6): 2650–2655.

Sheskin, David J. 2003. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, Inc.

Sutton, R. S., and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge University Press.

Thathachar, M. 1987. "Learning Automata with Changing Number of Actions." *IEEE Transactions on Systems, Man, and Cybernetics* 17 (6): 1095–1100.

Thathachar, M. A. L., and P. S. Sastry. 2004. *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Kluwer Academic Publishers.

Torkestani, J. A. 2012. "Degree-Constrained Minimum Spanning Tree Problem in Stochastic Graph." *Cybernetics and Systems* 43 (1): 1–21.

Torkestani, J. A., and M. R. Meybodi. 2012a. "A Learning Automata-Based Heuristic Algorithm for Solving the Minimum Spanning Tree Problem in Stochastic Graphs." *The Journal of Supercomputing* 59 (2): 1035–54.

———. 2012b. "Finding Minimum Weight Connected Dominating Set in Stochastic Graph Based on Learning Automata." *Information Sciences* 200 (1): 57–77.

Torkestani, J., and M. R. Meybodi. 2010. "Learning Automata-Based Algorithms for Finding Minimum Weakly Connected Dominating Set in Stochastic Graphs." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 18 (06): 721–758.

Torkestani, Javad Akbari. 2013. "A Learning Automata-Based Algorithm to the Stochastic Min-Degree Constrained Minimum Spanning Tree Problem." *International Journal of Foundations of Computer Science* 24 (03): 329–348.

Vafashoar, Reza, and Mohammad Reza Meybodi. 2016. "Multi Swarm Bare Bones Particle Swarm Optimization with Distribution Adaption." *Applied Soft Computing* 47: 534–552.

Vongsingthong, Suwimon, Sirapat Boonkrong, Mario Kubek, and Herwig Unger. 2015. "On the Distributions of User Behaviors in Complex Online Social Networks." In *Recent Advances in Information and Communication Technology 2015*, 237–246. Springer.

Wang, Mengmeng, Wanli Zuo, and Ying Wang. 2016. "An Improved Density Peaks-Based Clustering Method for Social Circle Discovery in Social Networks." *Neurocomputing* 179: 219–227.

Wasserman, S. 1994. *Social Network Analysis: Methods and Applications*. Cambridge University Press.

Watkins, Christopher John Cornish Hellaby. 1989. "Learning from Delayed Rewards." University of Cambridge England.

Watts, D.J., and S.H. Strogatz. 1998. "Collective Dynamics of 'small-World' Networks." *Nature* 393 (6684): 440–442.

Wilcoxon, Frank. 1945. "Individual Comparisons by Ranking Methods." *Biometrics Bulletin* 1 (6): 80–83.

Williams, Ronald J. 1988. *Toward a Theory of Reinforcement-Learning Connectionist Systems*. Northeastern University.

Wright, S. Paul. 1992. "Adjusted P-Values for Simultaneous Inference." *Biometrics*, 1005–1013.

Yan, Qiang, Lianren Wu, and Lan Zheng. 2013. "Social Network Based Microblog User Behavior Analysis." *Physica A: Statistical Mechanics and Its Applications* 392 (7): 1712–1723.

Zar, Jerrold H., and others. 1999. *Biostatistical Analysis*. Pearson Education India.

Zhang, Xiao-Yu. 2016. "Simultaneous Optimization for Robust Correlation Estimation in Partially Observed Social Network." *Neurocomputing* 205: 455–462.

Zhao, Yuxin, Wen Jiang, Shenghong Li, Yinghua Ma, Guiyang Su, and Xiang Lin. 2015. "A Cellular Learning Automata Based Algorithm for Detecting Community Structure in Complex Networks." *Neurocomputing* 151: 1216–1226.

Zhao, Yuxin, Shenghong Li, and Feng Jin. 2016. "Identification of Influential Nodes in Social Networks with Community Structure Based on Label Propagation." *Neurocomputing* 210: 34–44.

Zheng, Xianghan, Zhipeng Zeng, Zheyi Chen, Yuanlong Yu, and Chunming Rong. 2015. "Detecting Spammers on Social Networks." *Neurocomputing* 159: 27–34.

Zhong, Erheng, Wei Fan, Junwei Wang, Lei Xiao, and Yong Li. 2012. "Comsoc: Adaptive Transfer of User Behaviors over Composite Social Network." In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 696–704. ACM.

Zhu, Zhiguo, Jingqin Su, and Liping Kong. 2015. "Measuring Influence in Online Social Network Based on the User-Content Bipartite Graph." *Computers in Human Behavior* 52: 184–189.

About authors



Alireza Rezvanian received the B.Sc. degree in Computer Engineering from Bu-Ali Sina University of Hamedan, Iran, in 2007, and the M.Sc. degree in Computer Engineering with honors from Islamic Azad University of Qazvin, Iran, in 2010. He also received the Ph.D. degree in Computer Engineering at the Computer Engineering & Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2016. Currently, he works as a researcher in School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. Prior to the current position, he joined the Department of Computer Engineering and Information Technology at Hamedan University of Technology, Hamedan, Iran as a lecturer. He has authored or co-authored more than 60 research publications in reputable peer-reviewed journals and conferences including IEEE, Elsevier, Springer, Wiley and Taylor-Francis. He has organized various special sessions in international conferences in his area of expertise in Iran and abroad. He is a member of Technical Program Committees (TPCs) of various IEEE sponsored conferences in Iran and abroad. He has been a Reviewer of many reputable conferences and journals. He is also a Lead Guest Editor of Journal of Computational Science (Elsevier) for Special Issue on New Applications of Learning Automata-based Techniques in Real-World Environments. His research activities include soft computing, evolutionary algorithms, complex social networks, and learning systems.



Mohammad Reza Meybodi received B.Sc. and M.Sc. degrees in Economics from the Shahid Beheshti University, Tehran, Iran, in 1973 and 1977, respectively. He also received M.Sc. and Ph.D. degrees from the Oklahoma University, Oklahoma, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His current research interests include, learning systems, cloud computing, soft computing and social networks.