CFIN: A Community-based Algorithm for Finding Influential Nodes in Complex Social Networks

Mohammad Mehdi Daliri Khomami[1,*], Alireza Rezvanian[2,3], Mohammad Reza Meybodi[1] and Alireza Bagheri[1]

[1] *Department of Computer Engineering, Amirkabir University of Technology (Tehran Polytechnics), Tehran, Iran*

[2] *Department of Computer Engineering, University of Science and Culture, Tehran, Iran*

[3] *School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*

m.daliri@aut.ac.ir, rezvanian@usc.ac.ir, mmeybdi@aut.ac.ir, ar_bagheri@aut.ac.ir

Abstract

In influence maximization (IM) problem, as a fundamental algorithmic problem, is the problem of selecting a set of k users (refer as seed set) from a social network to maximize the expected number of influenced users (also known as influence spread). Due to the numerous applications of IM in marketing, IM has been studied extensively in recent years. Nevertheless, many algorithms do not take into consideration the impact of communities to influence maximization and some algorithms are non-scalable and time-consuming in practice. In this paper, we proposed a fast and scalable algorithm called Community Finding Influential Nodes (CFIN) that selects k users based on community structure, which maximizes the influence spread in the networks. The CFIN consists of two main parts for influence maximization: 1) seed selection; and 2) local community spreading. The first part of CFIN is the extraction of seed nodes from communities which obtained the running of the community detection algorithm. In this part, to decrease computational complexity effectively and scatter seed nodes into communities, the meaningful communities are selected. The second part consists of the influence spread inside communities that are independent of each other. In this part, the final seed nodes entered to distribute the local spreading by the use of a simple path inside communities. To study the performance of the CFIN, several experiments have been conducted on some real and synthetic networks. The experimental simulations on the CFIN, in comparison with other algorithms, confirm the superiority of the CFIN in terms of influence spread, coverage ratio, running time, and *Dolan-Moré* performance profile.

Keywords: Complex Network, Social Network Analysis, Influence Maximization, Community Detection.

1 Introduction

Many complex systems in different domains, such as social science, engineering, and biology, can be represented as networks. The idea of network representation leads to solving many real-life problems [1, 2]. In network model representation, nodes represent entities, and edges represent a specific relationship among nodes. A social network as a complex network is composed of a set of nodes as individuals and a set of edges representing particular relationships between individuals. A social network is organized based on communication among users, which makes a platform for any purpose, such as marketing. A popular way for marketing in a social network is viral marketing, which encourages customers to share information with their counterparts. Viral marketing [3] uses spreading mechanism through information diffusion in the network. In viral marketing, a subset of influential nodes is usually employed to spread the products/services to their family, friends, neighbors, or the contextual environment in which they operate. These spreaders propagate the products under a specific propagation model to others and finally produce the spread maximally [4], [5]. In other words, in viral marketing, a company may hope to spread the adoption of a new product from some initially selected adopters by the social links between users. In these scenarios, the core problem is how to detect the most relevant seed nodes. In this scope, different applications based on studying the results of diffusion process are achieved, including rumor diffusion [6], rumor control [7], network monitoring [8], revenue maximization [9] and social recommendation [10].

The problem of influence maximization (IM) is defined as finding a seed set of k users who could maximize the influence spread over a social network [11, 12]. In the last decade, considerable works have been carried out for influence maximization problem [6, 8, 10]. With the aid of humanities science, *Kempe et al.* in [13] proposed two well-known diffusion models, namely linear threshold model (LTM) and independent cascade models (ICM) and prove that influence maximization under both diffusion models is an NP-hard problem.

* Corresponding author. Email address: m.daliri@aut.ac.ir; Tel.: +98-21-6454-5120; Fax: +98-21-66495521

Recently, many algorithms have been proposed for influence maximization. In [13], a greedy algorithm is proposed based on hill-climbing to select k influential nodes. Moreover, it is proved that the greedy strategy forms an approximation guarantee to provide over 63% influence spread of the optimal seed set. *Leskovec et al.* in [14] introduced a greedy algorithm called Cost-Efficient Lazy Forward method (CELF) and exploit sub-modularity property in CELF algorithm to achieve near-optimal placements. In order to speed up the CELF, in [15] an extension called CELF++ is presented, which reduces the number of iterations significantly. *Kundu et al.* [16] investigated two steps targeted set selection problem and proposed a deprecation based greedy algorithm. The first part of their algorithm is the estimation, which evaluates the performance of each node while the second step is marking that dropped nodes in later iterations. *Zhou et al.* [17] proposed upper bounds to degrade the number of Monte-Carlo simulations in greedy-based algorithms. The algorithm introduced a new upper bound based lazy forward algorithm (UBLF) to produce seed nodes. *Song et al.* [18] investigated the influential node tracking (INT) problem as an extension of traditional IM. The main goal of the INT algorithm is to detect some influential nodes and maximize the influence during the network evolves, and a greedy algorithm, i.e., upper bound interchange, was introduced. In [19], *Chen et al.* used a degree discount heuristics algorithm that significantly decreases the running time of the greedy algorithm by considering the direct influence of a node to one-hop neighbors. Also, [20] proposed a dynamic selection algorithm for a given item based on real-world diffusion traces and online relationships. Based on the algorithm, first, the affected users by diffusion are divided into different communities based on different topics and ranks. Then, top–k influential users are selected for a given topic. *Zho et al.* in [21] developed an algorithm based on the coloring problem in order to find influential nodes. In their algorithm, the nodes with the same color are sorted into an independent set, and then based on a given centrality measure, the nodes with the highest centrality in an independent set are selected.

On the other hand, *kim et al.* in [22] investigated a scalable influence approximation algorithm, namely independent path algorithm (IPA), and used influence path for approximating influence in the network. *Kim et al.* [23] proposed an extension model of independent cascade with an interfering time restriction and called continuously activated and time-restricted (CT-IC). In this model, each active node has many chances to the inactive neighbor. The process of activating nodes is continued until reaching a given time. A polarity-related algorithm for influence maximization (PRIM) in a signed social network is proposed in [24], which aims to find the set of seed nodes with maximum positive influence or maximum negative influence. According to the PRIM model, they defined two sub-problems, including positive influence maximization (PIM) problem and negative influence maximization (NIM) problem, and established a greedy algorithm to achieve an approximation ratio of $\mathbf{1} - \mathbf{1}/e$. In [25], a greedy algorithm is devised on a small subset of nodes consisting of the top nodes as ranked by the PageRank algorithm on the social network. In [26] the shortest-path problem is considered, and a novel algorithm based on influence cascade models is outlined for finding the shortest path.

Moreover, efficient algorithms to compute the influence spread under these models are provided. *Ohsaka et al.* in [27] investigated the existence of a hub in social networks in order to accelerate breadth-first searches. Also, they guaranteed theoretically that the proposed algorithm reduces the number of necessary simulations to select a seed set accurately. *Goyal et al.* [28] worked on influence models and proposed the sim-path algorithm, which tries to estimate activation probability by examination paths that exist between seed nodes and other nodes in the network. In [29], another algorithm is introduced for further relaxation of influence maximization in a directed acyclic graph (DAG) with linear time complexity. The proposed algorithm assumed that each node could influence only a certain number of local neighbors; therefore, they proposed a local directed acyclic graph LDAG to solve IM. In [30], an efficient algorithm to compute the Expected Influence Spread (EIS) is outlined. They computed EIS of a node by finding cycles through it and applied an exact algorithm to compute EIS among a small number of hops. Then, an approximation algorithm is proposed to estimate EIS without the hop constraint. Based on the proposed algorithms, they finally developed an efficient greedy based algorithm for influence maximization. An improved greedy algorithm (SMG) is presented in [31], in which graph instances needed for Monte-Carlo simulation are restricted. Additionally, they showed that the proposed algorithm improved time complexity. In [32], a heuristic approach named shapely value is outlined for the LT model based on the heuristic SPIN. In this algorithm, the information corresponding to the diffusion process is modeled as a cooperative game, and by using the shapely values of the nodes, their network values are computed. They also showed the computational efficiency of the proposed algorithm by experiments.

Community structure [33, 34] is one of the essential characteristics of networks, which is defined as densely connected among community members, and sparsely connected with the rest of the network [35]. In the real world, people are not only present in the network, but also they are in different communities. So it is reasonable and inevitable to regard the community structure to identifying influential nodes. By considering the fact that people belong are form communities, *Li et al.* [36] investigated a community-based algorithm called conformity aware influence maximization (CINEMA). CINEMA presents conformity in the IM process primarily. Based on conformity, *Li et al.* described two different diffusion models as, conformity aware cascade and context-based conformity aware cascade. CINEMA identifies communities and selects seed nodes from different communities by computing their marginal gain. The authors also propose a distributed framework for CINEMA. *Guo et al.* [37]

investigated a cluster-based algorithm that breaks the entire network into some communities. This algorithm maximizes the influence across trajectory databases. The policy of the algorithm is different from IM approaches and does not consider any diffusion model for influence propagation. *Li* et *al.* [38] presented an algorithm that is regarding the user's interest in different areas. And also, they present a framework to estimate IM based explicitly on query processing and tries to enhance the effectiveness of seed. In [39], the authors introduced a heuristic approach to maximize the influence spread in the network. This algorithm selects boundary nodes as seeds randomly and then operates seed tuning based on the friendship paradox to improve the quality of seeds. *Wilder et al.* [40] suggested two steps community-based algorithm named approximating random walks to influence a socially explored network *(ARISEN)*. The First step of *ARISEN* is initializing the weight of each community, and the second step gives a weight refinement process to maximize influence.

In this paper, we focus on a fast and scalable algorithm for finding high-quality seed set in social networks. The proposed algorithm utilizes a community detection algorithm to reveal different communities. Then it constructs a new network by selecting each of the communities as a node and computes the number of connections inside of communities and assigns this number as the density of the edge created in the new network, then selects important communities from the new network. We note that the policy of selecting initial nodes in the new network is the same as selecting important communities from the initial input network. The algorithm chooses final seeds after evaluating the initial seeds and computes their importance with other important nodes. The experiment results demonstrate that the proposed algorithm is superior in terms of efficiency and scalability in comparison with baseline algorithms.

The rest of the paper is organized as follows. Section 2 presents some backgrounds for the information diffusion model, influence maximization, and community detection algorithms. The proposed algorithm called CFIN is described in section 3. Experimental results are presented in section 4, and section 5 concludes the paper.

2 Background

This section provides some basic description of information diffusion models, influence maximization (IM), and community detection algorithms, which are essentials in all influence maximization algorithms. The IM problem is defined on diffusion models to capture the information diffusion process among the users in an online social network. Consequently, we review diffusion models to construct the basis of the IM problem. Afterward, we formally define the IM problem and discuss the characteristics as well as the computational complexity of the problem under diffusion models.

2.1 SIM-PATH Model

The SIM-PATH [28] is a recent diffusion model for influence maximization, In *SIM-PATH* model, social network considered as graph $G = (V, E, b)$ where V is the set of nodes, E is the set of edges, and b is the weight of the edges, which has the range between zero and one that shows influence value. Initially, some nodes are activated, and others inactivated. *SIM-PATH* computes the spread of a node by summing the weights (*i.e.*, probabilities) of all simple paths starting from it. Therefore, the spread of a seed set S can be estimated by counting simple paths from nodes in seeds. Since the problem of enumerating all simple paths is NP-hard, the algorithm captures path around the small neighborhood where the size of the neighborhood can be controlled by the error. Also, *Goyal et al.* in [28] proposed an algorithm *SIMPATH-SPREAD*, which uses two optimization techniques for reducing influence spread estimation calls vertex cover optimization and Look-Ahead optimization. In the first iteration, vertex cover optimization significantly decreased the running time, while in subsequent iterations Look-Ahead optimization is used to reducing the number of backtrack iterations. This algorithm is efficient and scalable.

2.2 Influence Maximization

Influence maximization (IM) is a problem of selecting a set S of k users, whose influence spread over a social network is highest among any combination of k users. IM problem is formally defined as follows: Given $G = (V, E)$ where V is a set of nodes, E is the set of edges and budget k, it is defined as finding a set S containing k nodes, called a k-seed set, that maximizes the influence spread $\sigma(S)$ over G.

$$S = argmax_{S \subset V, |S| = k} \sigma(S) \tag{1}$$

2.3 Community Detection

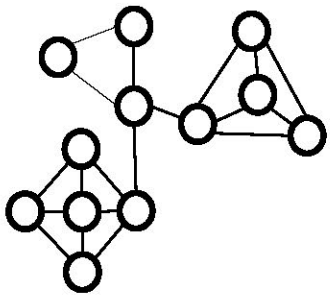
Community structure is defined as a subset of people who have a high interaction with each other. Edges of a community are dense, where edges between communities are sparse. Community detection can be formulated as a clustering problem that splitting nodes of graph G into P subset of $C_1, C_2, \ldots, C_p$ so that union of them is equal to the total of V [41]. For a comprehensive overview of widely used methods and techniques, one can view works performed in [42–44]. Since community detection is an essential part of our proposed algorithm, hence we will review the related works in this section in brief.

- **Clique percolation**. Clique percolation implies that edges in the same community capable of producing a clique, but edges in the different communities are incapable of forming a clique. These algorithms are successful and efficient for the interpretation of community structures. They can also discover strongly connected sub-graphs to estimate the actual communities. Several algorithms are presented which use the percolation method, including (Kumpula et al., 2008; Z.-G. Luo et al., 2011; Palla et al., 2005).
- **Label propagation**. The mechanism of the label propagation algorithms (LPA) is assigning different labels to each node based on particular rules. This process is continued until the nodes with the same label are grouped in the same community. The LPA algorithm was first utilized in the community detection domain [45]. Concerning the problems existing in the original LPA, several enhanced algorithms were introduced based on the naive algorithm. COPRA algorithm [46] and SLPA algorithm [47] were proposed to find overlapping communities. For the large scale networks, variation of the LPA algorithm, such as balanced label propagation [48], the multivariate graph-based method [49], AntLP [50], and parallel SLPA[51], was proposed.
- **Link partitioning.** The link partitioning-based method was introduced in [52] and used by [53] for large scale networks. The intuition behind this algorithm is, a link in the network usually describes the unique association; the link-based algorithms initially cluster on edges of the network and then map the link communities to node communities by collecting nodes occurrence to all edges within each link communities.
- **Local expansion and optimization**. Local expansion and optimization algorithms operate based on enlarging original communities or a partial community. In general, these methods concentrate on a local profit function, which tries to find the nodes with high connectivity. In [54], proposed the Greedy Clique expansion algorithm (GCE). The algorithm first detects the largest clique as an initial seed, and then around these seeds, an objective function was optimized locally with the aid of greedy search strategies. In [55], introduced a local community method to produce community expansion. The method selects a group of nodes instead of a single node as seed and supplies the local information of neighboring nodes as input. Also, using the neighboring group can distinguish a proper community for a hub node.
- **Agent-based and dynamic algorithms.** These algorithms combine cooperation and competition among particles to generate a fuzzy output for each node in the network. The fuzzy output indicates the levels of association of the nodes to each group of communities. Then, an overlapping measure is determined from these fuzzy outputs to show the confidence level of the output label. *Badie et al.* [56] presented an agent-based algorithm called a swarm of agent's teamwork for overlapping community identification (SATOCI) that uses a number of agents for investigation of the input network. These agents view different nodes' closeness in their decision to form communities.

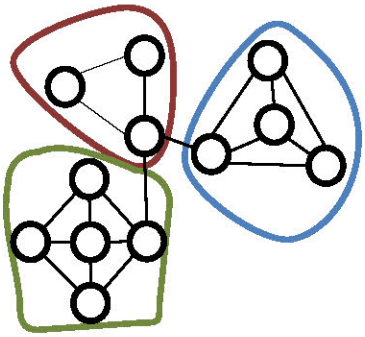
We note that our proposed influence maximization algorithm is capable of replacing by any community detection algorithms. However, we have used a distributed learning automata-based algorithm for community detection (DLACD), which is fast, scalable, and its quality of extracted communities is proper. The DLACD is one of the learning automata-based algorithm presented by *Khomami et al.* in 2016 [57]. In the proposed algorithm, each vertex of the network is isomorphic with learning automata. Based on the cooperation among a network of learning automata and updating action probabilities, the algorithm interactively tries to identify high-density local communities and improve its performance. The details of this algorithm are presented in the following section.

3 CFIN Algorithm

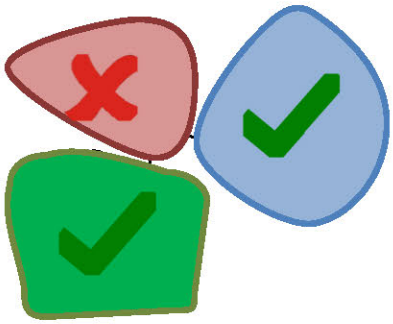
In this section, we describe the proposed algorithm for selecting top-k influential nodes based on community structure for influence maximization in social networks. It is assumed that the input network $G = \langle V, E \rangle$ is undirected, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E \subseteq V \times V$ are the set of nodes and edges in the input network, respectively. With the aid of community detection, the input network is divided into communities. Then, the most appropriate communities are selected with respect to the size and density of communities. Moreover, some nodes are selected as candidate nodes from the proper communities. A common choice for selecting these nodes is the maximum degree. Finally, we select *top-k* nodes from selected nodes in the last step by reviewing the initial seeds and compare their importance with other important nodes for influence maximization. In the rest of this paper, we called the proposed algorithm as *CFIN* (Community Finding Influential Nodes). The main steps of the CFIN are illustrated in figure 1.



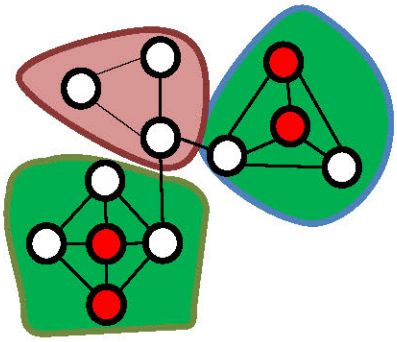
Step 1. Input network with 12 nodes and 3 communities.



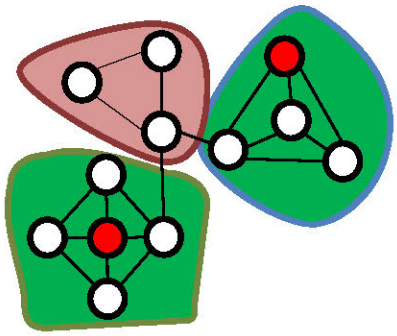
Step 2. Extracted three communities by community detection algorithm



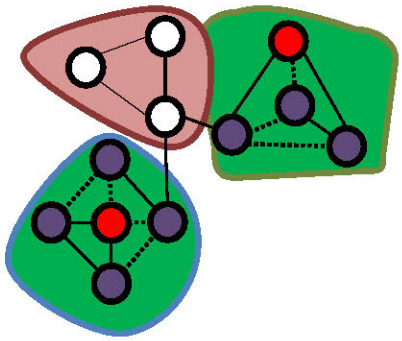
Step 3. Selected two communities colored as blue and green with a green tick as candidate communities and ignored the other community colored as red with a red cross



Step 4. Selected candidate seed set nodes colored as red for each selected community



Step 5. Finalized the selected seed set nodes colored as red for each community



Step 6. Resulted influence spread of final activated nodes colored as dark blue influenced by the selected seed set nodes colored as red

Figure 1. A toy example for steps of the CFIN algorithm for influence maximization

3.1 Steps of CFIN algorithm

The general steps of the CFIN Algorithm, which consist of 5 steps, including 1) Community Detection 2) Candidate Community Selection, 3) Candidate Node Selection, 4) Initial seed Selection, and 5) Final Seed Selection. In the following, we describe in detail the above five steps. To initialize the algorithm, we consider some basic assumptions. In the community detection step, the community detection algorithm is used, and candidate communities are selected. Since a community, which holds seed may affect the quality of influence spread. Thus, in the CFIN algorithm, there is a criterion in which the proper communities are detected. One of the most important properties for an appropriate community is selecting communities based on their size. Hence, for seed selection, it makes sense to choose the seed nodes from the communities with greater cardinality. Then in order to refine the selected seeds, communities with cardinality larger than the average size of the k communities are selected. Based on the proposed idea, in case we select more than one seed node from a community, a node might be selected as a seed repeatedly. Moreover, there is no guarantee that the selected seed nodes have a proper impact on the influence spread. To overcome these drawbacks, after selecting k communities, we substitute a portion of selected seed nodes with the best possible choices that are not selected yet. A typical property for selecting the best possible choices is the density of the community with a maximum degree. This policy for selecting seed nodes leads to an expand influence spread in the obtained solution properly. Figure 2, as an Algorithm 1, shows the details of the proposed algorithm called CFIN.

Algorithm 1: CFIN (G, k)

Input: Graph $G = (V, E)$ and k
Output: S Seed set
Step 1: Community Detection
CD = Detecting communities of G using Distributed Learning Automata Based Algorithm for Community Detection (DLACD);
Step 2: Candidate Community Selection
CandidateCommunitySelection ← *CommunitySelection(CD);*
Step 3: Candidate Node Selection
$MeanCommunitySize = \frac{\sum_{i=1}^{k} |C_i|}{k}$;
$SelectedNodeQuata = \left\lceil \left| \frac{N_i}{N_i - MeanCommunitySize} \right| * \alpha \right\rceil + \beta$;
ImportantNodesPool = Selecting important Nodes from each community by degree centrality measure based on *SelectedNodeQuata*;

Step 4: Initial Seed Selection
Seed set = Selecting top nodes from *ImportantNodesPool* from each community based on *SelectedNodeQuata;*
Step 5: Final Seed Selection
Final-Seeds= SeedRefinment (G, Seed set, k, CD, ImportantNodesPool);
End Algorithm

Figure 2. Pseudo-code for the CFIN algorithm

Now, we can describe CFIN algorithm by explaining the details of each step. Initially, the input graph and number of required seed nodes are given to the algorithm.

3.1.1. Step 1. Community Detection

In order to decrease the number of nodes that must be considered as seeds and to reduce the execution time of the algorithm, the first step consists of finding communities in the networks. In this paper, we have used a novel algorithm based on learning automata for community detection problem due to scalability and does not need to know any prior knowledge of the number of communities. In the DLACD algorithm, a community is recognized as a group of nodes with dense interconnections and sparse interconnections with nodes outside of it. According to the definition of communities, the proposed algorithm tries to find communities iteratively and select appropriate communities that minimize an objective function. A common choice for the objective function is supposed to be normalized cut. It is important to note that our community detection works on undirected networks and the networks used here are directed, so we treat direct networks as same as undirected by omitting the direction.

3.1.2. Step 2. Candidate Community Selection

In this step, we select some candidate communities from those obtained from the community detection algorithm as figure 3. To this end, since we may have many choices to involve nodes in the diffusion process, we sort the communities based on size. We define the size of the community as the number of included edges. Note that if we need to select k nodes for maximization, we have to select k biggest communities. After selecting communities, the nodes within communities are ranked based on their degree centrality. Intuitively, a node has a high degree of centrality value if it is located between many other nodes in the network. Since the nodes in the center of communities have more connections with others, more people can be convinced to be adopted with them.

Algorithm 2. Important Communities Selection Algorithm
Input: Graph $G = (V, E)$, Community Detection Algorithm (*DLACD).*
Output: Important Communities.
CD = Detecting communities of G using Distributed Learning Automata Based Algorithm for Community;
Detection (DLACD);
CommunitySize= Compute each community size based on Equation(2);
SortedCommunities = Sort Communities of G in descending order based on CommunitySize;
ImportantCommunity = Select k communities from SortedCommunities + Hub nodes;
SelectedNodeQuata= Select top 10% of nodes from each ImportantCommunity based on Equation(3);
UpwardSortNode= Sorting SelectedNodeQuata in ascending order based on degree centrality;
DownwardSortNode= Sorting SelectedNodeQuata in descending order UpwardSortNode \ SelectedNodeQuata;
For (each community in ImporantCommunity) {
InfluenceUpward=Pop node v_i from UpwardSortNode and Compute ***Community-simpath*** (Community, v_i)
InfluenceDownward greater than InfluenceUpward then remove v_i from UpwardSortNode;
Add v_j to UpwardSortNode;
ImportantCommunity = add v_j community to ImportantCommunity}
End For
Return ImportantCommunity;

Figure 3. Pseudo-code for candidate community selection

In order to speed up the proposed algorithm, the first ten percent of the nodes with higher degree centrality is only selected as top sorted nodes from the result of the community detection algorithm. Also, to improve results, hub nodes with their communities are added to important nodes set, because the hub nodes have an important effect on influence spreading in the network. Selecting the hub nodes from the new network means selecting communities from the initial input network that has a central role as well as a connector among other communities. The goal is to increase the number of activated nodes in the input network. Hence some communities must be chosen as important communities that can affect the entire network.

3.1.3. Step 3. Candidate Node Selection

The main challenge in finding influential nodes is how to find seed nodes accurately. This phase, after selecting important communities and ranking nodes based on some criteria, some candidate nodes should be selected from each candidate community. It is necessary to point out that many criteria can be applied to select nodes from communities. Therefore, we have tried to study different criteria to study the impact on the initial candidate nodes.

In order to do so, we first compute the mean size of communities. Let C_i be the i^{th} community, the mean size of communities is equal to

$$Mean_c = \frac{\sum_{i=1}^{k} |C_i|}{k}, \quad (2)$$

where $|C_i|$ is the cardinality of the i^{th} community, and k is the total number of communities for selecting initial nodes. Moreover, after selecting the Mean community size, some nodes are selected as SelectedNodeQuata. Let *SelectedNodeQuata* be the set of nodes that have been selected as candidates from each community and defined by

$$SelectedNodeQuata = \left\lceil \left| \frac{N_i}{N_i - Mean_c} \right| \times \alpha \right\rceil + \beta , \quad (3)$$

where N_i is the number of nodes in the i^{th} community and α and β are the constant values for reinforcement of selection and minimum size of selection, respectively. The appropriate values for these constants depends on input network, but in our experiment are supposed to be $\alpha = 10$, and $\beta = 2$.

3.1.4. Step 4. Initial Seed Selection

After the candidate node selection step is done, the initial seed selection step starts. In order to find initial seeds set, at first, the important nodes and hubs are sorted based on the degree in ascending and descending order. At this point k, initial seeds are selected or exchanged and examined to determine whether they can maximize influence spread in the new network or not. To achieve this goal, we need to compare the influence spread in case a node is considered to be a seed or not.

The policy of selecting the k initial seed nodes varies according to applying different criteria to choose candidate nodes. For instance, in the Random selection, k nodes from communities are chosen randomly. Moreover, in maximum degree, the nodes are sorted in descending order, and then k high degree nodes are selected.

3.1.5. Step 5. Final Seed Selection

In the following, to refine the selected seed nodes in the fourth step of the proposed algorithm, we need to compare the influence spread in case a node is considered to be a seed or not. For this purpose, we describe computing influence spread based on the modified sim-path algorithm.

The local clustering coefficient of a node in a network quantifies how close its neighbors are to be a complete network. More precisely, the local clustering coefficient CC_i for a node v_i is given by the proportion of edges between the nodes within its neighborhood divided by the number of edges that could possibly exist between them. From a mathematical point of view, the local cluster co-efficient in an undirected graph is calculated by

$$CC_i = \frac{2|\{e_{jk}: v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)}, \quad (4)$$

where k_i is the number of neighbors of a node *i*, N_i is the set of $i's$ neighbors and e_{jk} indicates the edge between node v_j and v_k.

In the naive algorithm such a Sim-path [28], it is a common way to consider the parameter η as a pruning threshold to improve the quality of diffusion. However, since computing η is complicated, here to reach a significant diffusion, we use the average clustering co-efficient as a new criterion. The average cluster coefficient is defined as

$$\overline{C} = \frac{1}{n_c} \sum_{i=1}^{n_c} CC_i , \quad (5)$$

by n_c, we mean the number of nodes within the current community.

3.2 Computing influence spread

We use a modified sim-path as figure 4, which computed the diffusion spread inside communities for each initial active node. The sim-path is a lazy-forward algorithm presented in [28] in order to find a path. In the sim-path, all simple paths with less than a pre-defined length are discovered, and there is no constraint for searching space; however, in the modified sim-path, there is a constraint for searching simple path along with the community space. The modified sim-path is described in algorithms 2, 3, and 4. In the proposed algorithm, the seed set S, and pruning threshold based on the average community clustering coefficient Avg_{C_u} is given as input of the algorithm. For each member u of S, the BACKTRACK algorithm takes, in turn the node u, the pruning threshold η and a given set of nodes $W \in V - S + u$ as input.

Algorithm 2: ***Community-simpath***

Input: S as seed set,
Output: σ(S)
For each u∈ S **DO**
    σ(S) ← σ(S) + Backtrack (u, Community(u), Avg_c);
End For

Figure 4. Pseudo-code of *Simpath-community*

The *BACKTRACK* algorithm traverses all simple paths inside a community, starting with node u. For this propose, BACKTRACK uses a stack Q to maintain current nodes on the path, pp to keep the weight of the current path and spd to track the spread of the node u inside the community. Also, $D[x]$ preserve out-neighbors of x that has been detected so far. This process provides the possibility to explore all simple paths from a starting node inside communities. The main steps of the *BACKTRACK* algorithm are presented in Figure 5.

Algorithm 3. BACKTRACK Algorithm

Input: u, Community(u), Avg_c
Output: spd
Q ←{u}; spd ←1;
pp ←1;
D ← Null;
While (Q≠ ∅) **DO**
    [Q,D,spd,pp] ← Forward (Q, D, spd, Community(u), Avg c_u);
    u←Q.last(); Q←Q-u; Delete D[u]; v← Q.last();
    pp ← $\frac{pp}{b_{vu}}$;
End While

Figure 5. Pseudo-code of the *BACKTRACK* algorithm

In this algorithm, after variable initialization, the forward procedure is called consequently and traverses unvisited path up to now. Based on the *FORWARD* algorithm, the last node on a simple path x is selected and traverses the local community, which it belongs to as much as possible. In order to do this, a node y, which is adjacent to node x in the community but is not explored yet, is selected. If the new node y is added to the path, the weight of the path pp and the spread are being updated. Finally, the node y is added to the set of seen neighbors of x. The process is continued until no new node can be added. Figure 6 demonstrates the steps of the *FORWARD* algorithm.

Algorithm 4: ***FORWARD Algorithm***

Input: Q,D, spd, c_u, Avg c_u
Output: spd
x←Q.last();
While (∃ y∈N^{out} (x), $Coeff_y$ >=Avg c_u: y ∉ Q, y ∉ D[x], y∈ c_u) **DO**
    Q.add(y);
    pp←pp.b_{xy}; spd←spd+pp;
    D[x].insert(y); x← Q.last();
End While

Figure 6. Pseudo-code of the FORWARD algorithm

3.3 Complexity analysis of the CFIN

In this section, we aim to analyze the time complexity of the CFIN Algorithm. The CFIN consists of 5 steps, so its time complexity is the sum of the time complexity of these five steps. In the first step, the CFIN uses the DLACD algorithm for community detection. The worst computational cost in the DLACD includes the cost of the normalized cut as an objective function in which its time complexity is equal to $O(T \times n^2)$ where n is the number of network nodes and, T is the number of iterations. Since T has a constant value, thus, the total complexity of DLACD is equal to $O(n^2)$. The second step consists of the candidate community selection, which includes sorting communities and selecting relevant communities, takes $O(|C|)$, where C is the number of communities in the network. In the third step, after selecting important communities and ranking nodes, some candidate nodes should be selected from each community, which is $O(\alpha \times |C|)$, where α is a constant factor. In the fourth step, the CFIN selects k nodes and

compare seeds or not from the communities. The time complexity of the fourth step is $O(n \times \log(n))$ due to the sorting. In the fifth part, the seed refinement is $O(n)$ because it is linear.

4 Experimental results

In this section, the performance of the CFIN is investigated on several real networks including Zachary's Karate Club network denoted as *Karate* [58], Bottlenose Dolphins network denoted as *Dolphins* [59], American College Football network denoted as *Football* [60], Political Blog network denoted as *Polblogs* [61], co-authorship network of High Energy Physics (Theory) arXiv denoted as *NetHEPT* [62], and Epinions social network denoted as *Epinions* [63], all of these networks known as a set of well-known benchmark network dataset and also a set of synthetic networks including *Erdős–Rényi* model denoted as *ER-1000* [64], Watts–Strogatz model denoted as *WS-1000* [65], and *Lancichinetti–Fortunato–Radicchi* benchmark model denoted as *LFR-5000* [66]. The characteristics of both real and synthetic networks used for experiments given in Table 1. For synthetic networks, we use the well-known random graph, *Erdős–Rényi* model (ER model). Another synthetic network is the small-world network of the *Watts-Strogatz* model (WS model) that shows some properties of real-world networks such as small average path length and large clustering coefficient. Also, *Lancichinetti–Fortunato–Radicchi* benchmark model (LFR model) is another synthetic modular network in particular for testing community detection algorithms.

We set network parameters to $N = 1000$ and $p = 0.2$ for the ER model, where N and p represent the number of nodes and probability of including edges in the network, respectively. For the WS model, we set $N = 1000$, $k = 4$, and $p = 0.2$, where N is the number of nodes, k is the degree of an initial k-regular graph, and p is the re-connecting probability of edges. For the LFR model, parameters are set as $N = 5000$, $k = 15$, $max_k = 50$, $min_c = 50$ and $max_c = 150$, and $\mu = 0.2$, where N denotes the number of nodes, k denotes the average degree, max_k denotes the maximum degree, min_c denotes the minimum size of the community, max_c denotes the maximum size of the community and μ denotes the mixing parameter of the topology. Also, we assume that all undirected networks are double weighted directed for influence maximization. To calculate edges weight ($b_{u,u}$) for all networks, we use the same method that was applied in [67] as follows:

$$b_{u,v} = \frac{A(u,v)}{N(v)}, \tag{6}$$

where $A(u,v)$ is the mutual actions of u and v, and $N(v)$ is a normalized factor that is calculated as follows:

$$N(v) = \sum_{u \in N^{in}(v)} A(u,v), \tag{7}$$

where $N^{in}(v)$ is the set of nodes that has an edge to v.

Table 1. Description of the test networks used for experiments.

Network	*Node*	*Edge*	*Description*
Dolphins	62	159	Bottlenose dolphins network [59].
Epinions	75K	508K	Epinions social network of a general consumer review site [63].
ER-1000	1K	1k	Synthetic random network of Erdős–Rényi model [64].
Football	115	615	American college football networks [60].
Karate	34	78	Zachary's karate club network [58].
LFR-5000	5K	38K	Synthetic community based network of Lancichinetti–Fortunato–Radicchi benchmark model [66].
NetHEPT	15K	62K	Co-authorship network of High Energy Physics (Theory) arXiv [62].
Polblogs	1.5K	19k	US political blog network [61].
WS-1000	1K	1k	Synthetic small-world network of Watts–Strogatz model [65].

4.1 Criteria for Selecting Initial Active Nodes

Here we recall some well-known criteria for initial active node selection such as *Betweenness* centrality, *Closeness* centrality, and *Eccentricity* centrality. Our next experiment is conducted to investigate the effect of different criteria on the diffusion process.

One of the most important criteria for selecting initial active nodes is *Betweenness* centrality, which is calculated for a specific node v by computing the number of the shortest path passing through node v and is formulated as

$$Betweenness(v) = \sum_{i \neq j, i \neq v, j \neq v} \frac{g_{ij}(v)}{g_{ij}}, \tag{8}$$

where $g_{ij}(v)$ indicates the total number of shortest paths from node i to node j that pas through node v, and g_{ij} denotes the total number of the shortest path from node i to node j in the network.

Closeness centrality is known to be other proper criteria for initial active node selection. The closeness centrality of a node, as defined in equation (9), is the inverse of the sum of the length of the shortest paths from node v to all other nodes in the network and defined by

$$Closeness(v) = \frac{1}{\sum_{i \neq v} d_{vi}}, \tag{9}$$

where d_{iv} is the length of the shortest path between node v and node i.

Moreover, eccentricity has been noted to be appropriate criteria. For a specific node in a connected network, it is the maximum length of the shortest path to other nodes in the network.

4.2 Baseline Algorithms

Here, we provide a summary of the five well-known baseline algorithms for investigating their performances in comparison with the performance of the CFIN algorithm. Table 2. is the description of the baseline algorithms.

- **Rand:** In this method, a node is taken at random from the network and placed in the seed set.
- **LDAG:** The LDAG algorithm, which is presented in [29], estimates the spread of each node in its local DAG (Directed Acyclic Graph) and provides proper results in the quality of seed nodes.
- **MC-CELF:** The greedy approximation algorithm with approximation ratio $(1 - \frac{1}{e} - \epsilon)$ for any $\epsilon > 0$, to calculate the spread of a node. In this algorithm, Monte-Carlo (MC) simulations were conducted 10,000 times to calculate the influence spread of the seed sets [14].
- **CELF++:** The CELF++ algorithm is an improvement version of CELF, which overcomes the number of iterations of the algorithm considerably [15].
- **CIM:** This algorithm uses the heat diffusion model to influence maximization. The algorithm initially, discovers communities with the aid of a hierarchical community detection algorithm called *Hclustering*, then it selects several communities as significant communities based on their structural characteristics, and finally, candidate nodes from these critical communities are determined [68].
- **ComPath:** The algorithm first identifies the communities from the initial network. Next, using Betweenness centrality, a limited number of communities are selected. Then, some nodes will be selected as initial nodes. The algorithm examines different paths with different lengths, beginning from candidate nodes to pick final seeds [69].
- **PaS:** This algorithm works as a game-based model for influence maximization. In this algorithm, the strategy for selecting initial nodes is that the initial communities are selected based on their structure and size. Then, a game-theoretic approach is applied within communities to select initial nodes [70].

Table 2. The algorithms used in the experiments.

Algorithm	Description
Rand	A ranking algorithm which ranked nodes as random.
LDAG [29]	This algorithm is applied to the input network with the parameter h= 1.32 to control the construction of local DAG for each node.
MC-CELF [14]	The optimized version of CELF, in which the Monte-Carlo simulation is run for 10,000 times to obtain the final seeds.
CELF++ [15]	A faster and more efficient version of CELF.
CIM [68]	Community-based influence maximization in social networks.
ComPath [69]	A fast algorithm for finding the most influential people based on the linear threshold model.
PaS [70]	A game-theoretic approach for seeding a subset of nodes.

As a first step, we have to determine the number of selected communities and candidate nodes that are selected from each community. To this end, different experiments have been conducted. All experiments for the algorithm were launched on a system with a hardware configuration of Intel® Core i5 2.67 GHz and 4 GB RAM.

4.3 Experiments

4.3.1. Experiment I

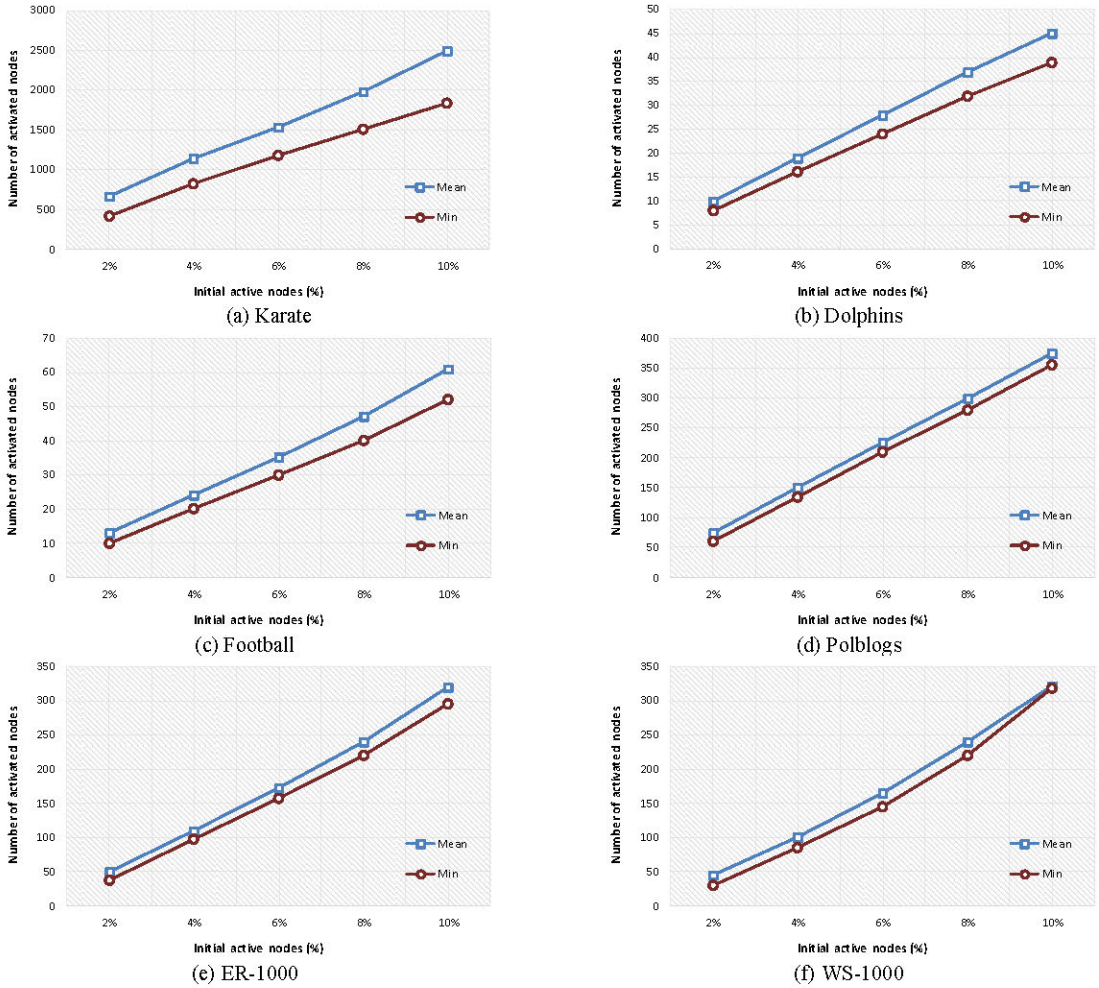
This experiment is carried out to study the performance of the CFIN algorithm in terms of the number of activated nodes at the end of the diffusion process. To compute the number of activated nodes, we performed two

different strategies for detecting the effect of the clustering coefficient. First, we considered that the average value of the clustering coefficient to decide whether the obtained number of activated nodes is effective or not. However, the minimum value of the clustering coefficient comes into use as a lower bound for influence spread. To perform this experiment, we plot the number of activated nodes versus initially active nodes for different networks in figure 7. We select a different number of initial active nodes from the input network with varying size from 2% to 10% of nodes with step length 2% of nodes. We note that the result corresponding to a higher amount of initially activated nodes is neglected since they did not contain any impressive achievement. The initial active nodes are selected based on the maximum degree inside communities. Considering the obtained results, we may conclude the following statements:

(i) The number of activated nodes at the end of the diffusion process is affected by the structure of communities and their size. If the structure of the network graph consists of a few numbers of balanced communities, the influence spread based on the minimum clustering coefficient outperforms the results obtained by the mean clustering coefficient. On the other hand, the execution time spent by the minimum clustering coefficient is higher; hence, it makes sense to use the mean clustering coefficient and improve the efficiency. To do so, we conduct experiment III to investigate the effect of the policy based on which the initial active nodes are selected.

(ii) The number of initially activated nodes plays a vital role in the quality of influence spread, as higher as this value is, the resulting number of activated nodes increases. However, we did not observe any outstanding improvement for a higher percentage of initial activated nodes than 10% of nodes.

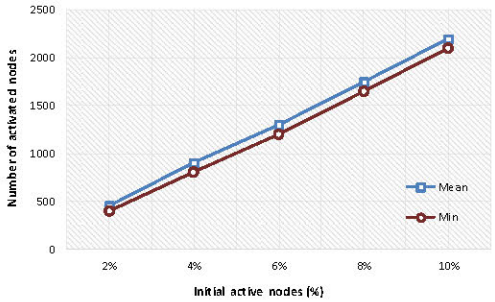
(iii) On the other hand, the policy of selecting initially active nodes may have an impact on the diffusion process from two different points of view: the trend and the number of final active nodes.



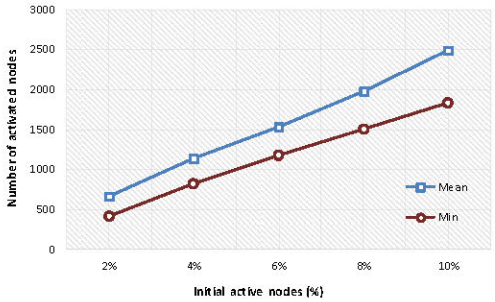
(a) Karate (b) Dolphins

(c) Football (d) Polblogs

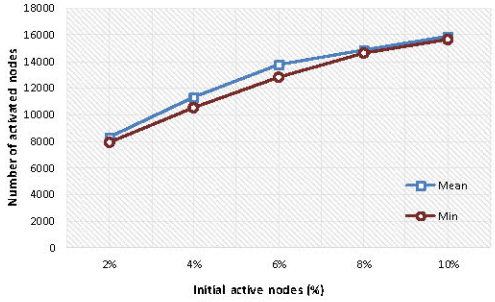
(e) ER-1000 (f) WS-1000



(g) LFR-5000



(h) NetHEPT

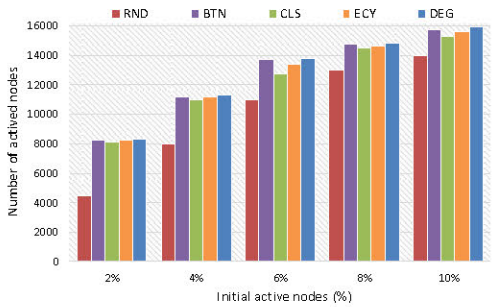


(i) Epinions

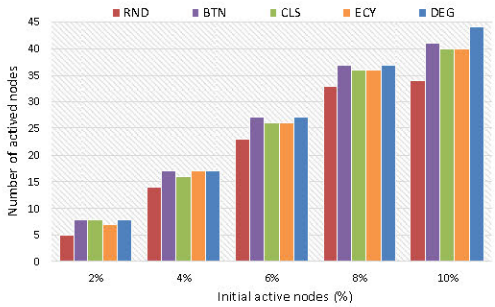
Figure 7. Comparison of the minimum and mean values for the selection of initial active nodes.

4.3.2. Experiment II

This experiment is conducted to study the impact of the initial node selection via selecting different seed nodes such as random node selection denoted as RND, betweenness centrality selection denoted as BTN, closeness centrality selection denoted as CLS, eccentricity centrality selection denoted as ECY, and maximum degree centrality selection denoted as DEG with different sizes. The comparison is made on the number of affected nodes at the end of the diffusion process. The results of this experiment for different number of initial active nodes from the input network with varying size from 2% to 10% of nodes with step length 2% of nodes are illustrated in Figure 8 for both real and synthetic networks. From the obtained results, it can be seen that in all networks, BTN and DEG achieved a good influence spread results than that of other algorithms. Also, a reason for BTN leading to a proper result is that the nodes with large inside community degrees that are not selected yet have more chance to be chosen as initially activated nodes. We also note that adding more initial active nodes caused the number of affected nodes to increase sharply, and the number of active nodes is gradually increased. Moreover, considering the obtained results, it may conclude that the effect of different strategies for selecting initial nodes on the diffusion process is more significant for real network datasets.



(a) Karate



(b) Dolphins

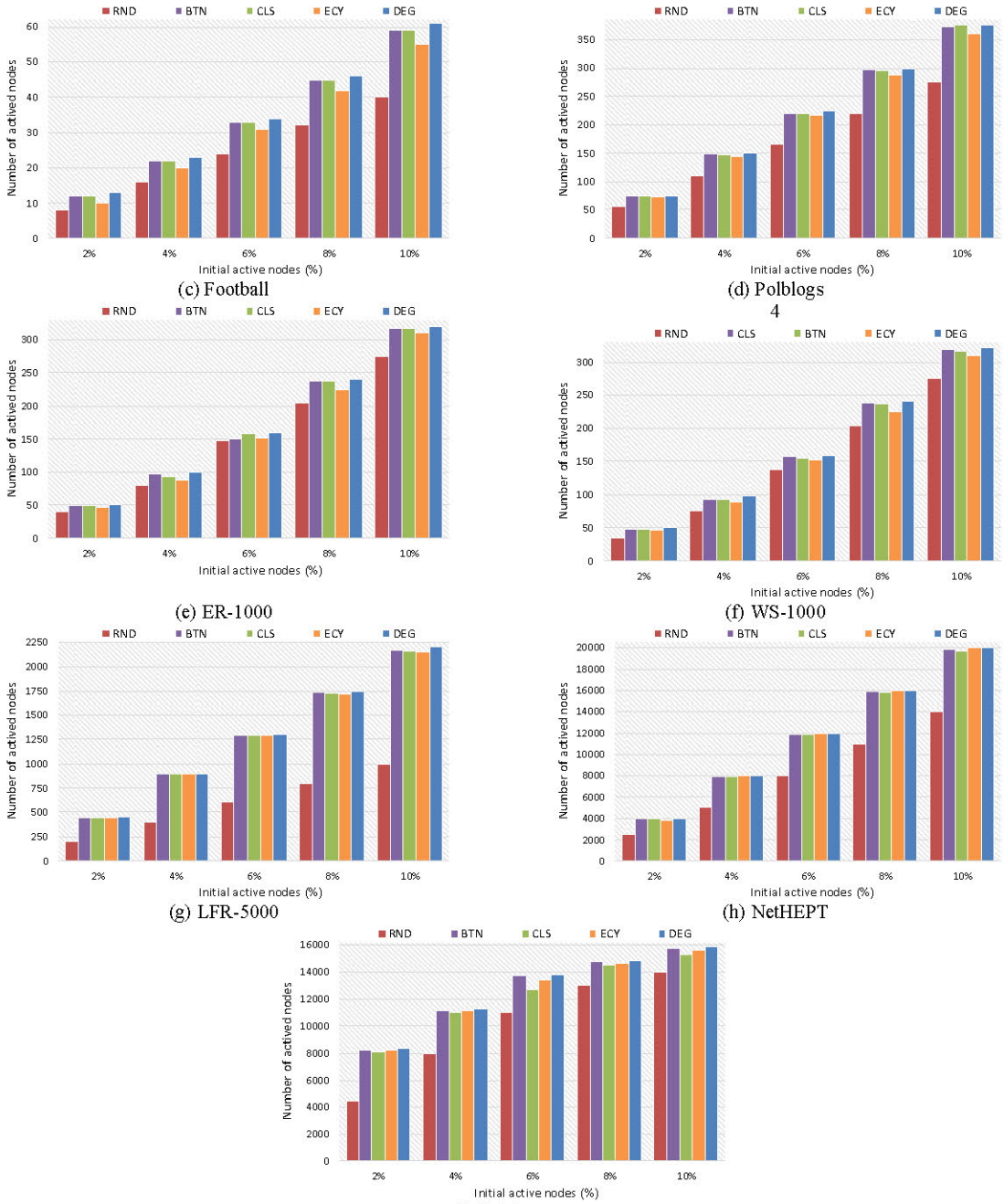


Figure 8. Comparison of the different centrality for selection of initial active nodes

4.3.3. Experiment III

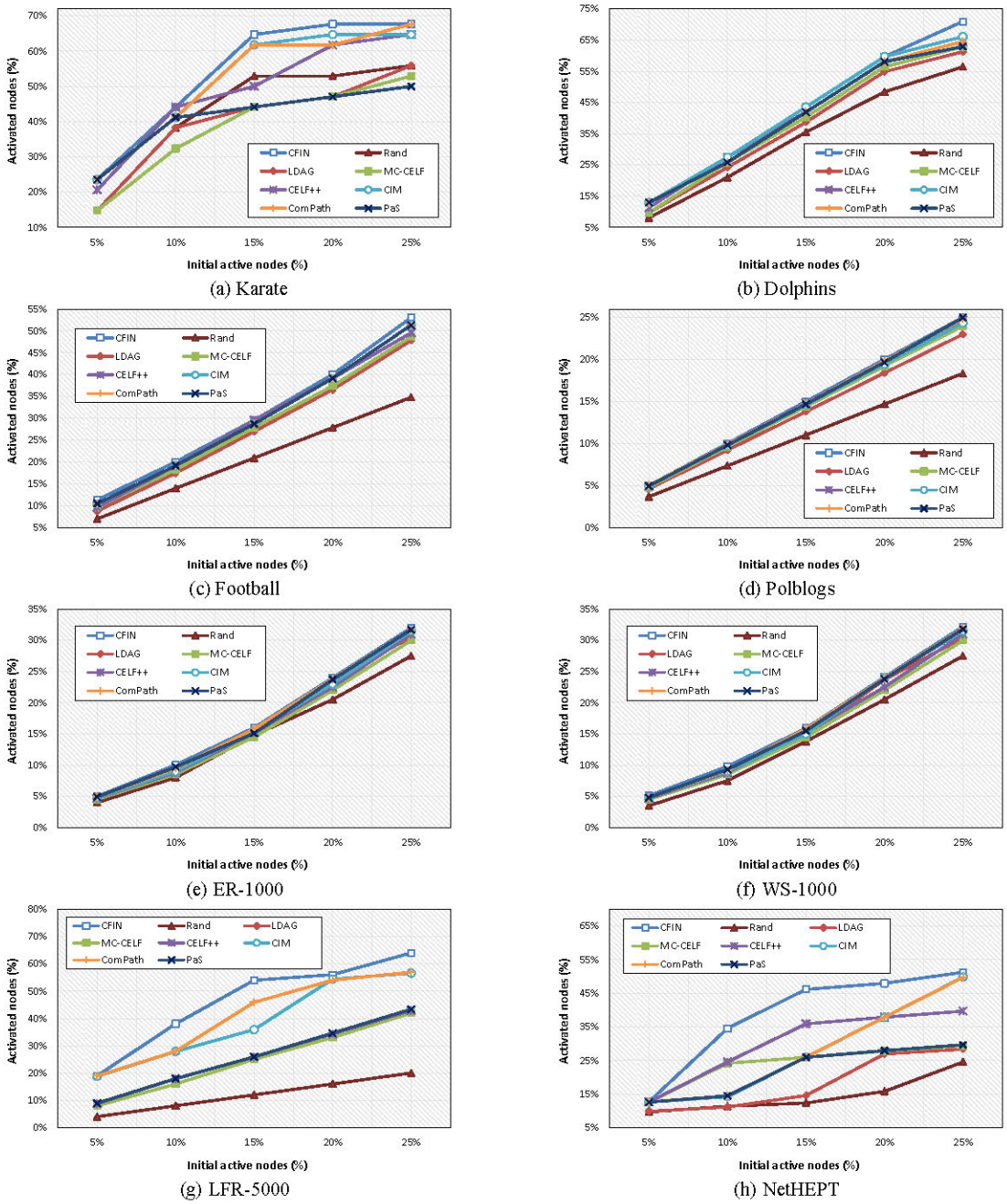
In this experiment, we aim to a make comparison of the number of activated nodes by the CFIN algorithm in comparison with other algorithms. In order to quantify the performance of the algorithms, we use the coverage ratio index as defined in equation (10)

$$Coverage\ ratio = \frac{\#\ Acivated\ nodes}{\#\ Total\ nodes}, \tag{10}$$

where $\#Activated\ nodes$ and $\#Total\ nodes$ correspond to the number of activated nodes at the end of the diffusion process by the algorithm and the total number of nodes in the network.

In this experiment, we evaluate the activated nodes at the end of the diffusion process for the CFIN and for comparing algorithms, including Rand, LDAG, MC-CELF, CELF++, CIM, ComPath, and PAS. In the Rand

algorithm, different seed nodes are selected randomly, and the average obtained result is reported; hence, it is not surprising that the Rand algorithm performs worse than all others. Moreover, the LDAG algorithm work based on constructing the local DAG (Directed Acyclic Graph) for each initial node. A more satisfying performance can be obtained by the use of a greedy algorithm named CELF. In the MC-CELF algorithm, a Monte-Carlo simulation is implied to estimate the influence spread of a seed set. To improve the functionality of MC-CELF, an optimization-based extension, CELF++, is constructed. CIM selects important communities based on community size, so if the difference average of community size is small, the algorithm may not outperform well. ComPath aims to reduce the number of checking graph nodes. Then some communities will be selected as candidate nodes and checks different paths with different lengths, starting from candidate nodes to select final seeds. When social networks have completely different internal structures, the better performance of the PaS reaches, however in some networks with large communities, it may fall its performance. Here to show the efficiency, we compare the coverage ratio for the CFIN algorithm and the seven mentioned algorithms in figure 9.

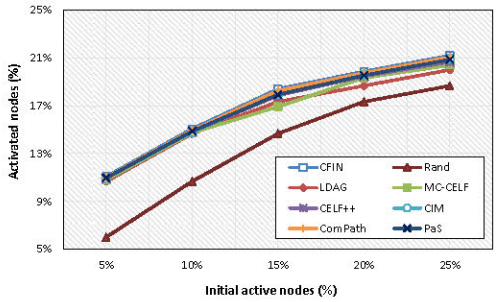


(a) Karate (b) Dolphins

(c) Football (d) Polblogs

(e) ER-1000 (f) WS-1000

(g) LFR-5000 (h) NetHEPT



(i) Epinions

Figure 9. influence spread with respect to coverage ratio in comparison with other algorithms

From the results presented in figure 9, it may be concluded that for some networks, the CFIN outperforms the other four algorithms. Even in such cases, there is no significant difference between the CFIN algorithm and other algorithms expects Rand. Additionally, in some other cases the best result corresponds to the CFIN algorithm.

Besides, we compare the performance of the CFIN and other algorithms in pairs as numerical representation with respect to growth ratio [71]

$$Growth\ ratio = \frac{A_{CFIN} - A_{ALG}}{A_ALG}, \tag{10}$$

where A_{CFIN} and A_{ALG} denotes to the number of activated nodes at the end of diffusion process by the CFIN algorithm and comparing algorithm in the network, respectively. The results of this experiment for different number of the seed set size in terms of the growth ration for Karate, Dolphins, Football, Polblogs, ER-1000, WS-1000, LFR-5000, NetHEPT, and Epinions are listed in Table 3 to table 9. In these tables, the last column in devoted to the average over from 5% to 25% sizes of seed set nodes. From the results one can see the superiority the CFIN.

Table 3. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for Karate

	5%	10%	15%	20%	25%	Average
Rand	0.60	0.15	0.22	0.28	0.21	0.29
LDAG	0.60	0.15	0.47	0.44	0.21	0.37
MC-CELF	0.60	0.36	0.47	0.44	0.28	0.43
CEF++	0.14	0.00	0.29	0.10	0.05	0.12
CIM	0.00	0.07	0.05	0.05	0.05	0.04
ComPath	0.00	0.07	0.05	0.10	0.00	0.04
PaS	0.00	0.07	0.47	0.44	0.35	0.27

Table 4. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for Dolphins

	5%	10%	15%	20%	25%	Average
Rand	0.60	0.31	0.23	0.23	0.26	0.33
LDAG	0.33	0.13	0.13	0.09	0.16	0.17
MC-CELF	0.33	0.06	0.08	0.06	0.13	0.13
CEF++	0.14	0.00	0.04	0.03	0.10	0.06
CIM	0.00	0.00	0.00	0.00	0.07	0.01
ComPath	0.00	0.06	0.04	0.03	0.10	0.05
PaS	0.00	0.06	0.04	0.03	0.13	0.05

Table 5. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for Football

	5%	10%	15%	20%	25%	Average
Rand	0.63	0.44	0.42	0.44	0.53	0.49
LDAG	0.30	0.15	0.10	0.10	0.11	0.15
MC-CELF	0.18	0.10	0.06	0.07	0.09	0.10
CEF++	0.18	0.05	0.00	0.02	0.07	0.06
CIM	0.08	0.05	0.03	0.02	0.03	0.04
ComPath	0.08	0.05	0.03	0.02	0.03	0.04
PaS	0.08	0.05	0.03	0.02	0.03	0.04

Table 6. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for Polblogs

	5%	10%	15%	20%	25%	Average
Rand	0.36	0.36	0.36	0.36	0.36	0.36
LDAG	0.09	0.09	0.09	0.09	0.09	0.09
MC-CELF	0.04	0.04	0.04	0.04	0.04	0.04

CEF++	0.01	0.01	0.01	0.01	0.01	0.01
CIM	0.01	0.03	0.03	0.03	0.03	0.02
ComPath	0.00	0.01	0.02	0.01	0.01	0.01
PaS	0.01	0.02	0.02	0.02	0.00	0.01

Table 7. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for ER-1000

	5%	**10%**	**15%**	**20%**	**25%**	**Average**
Rand	0.25	0.25	0.08	0.17	0.16	0.18
LDAG	0.04	0.11	0.08	0.01	0.05	0.06
MC-CELF	0.11	0.18	0.10	0.09	0.07	0.11
CEF++	0.06	0.14	0.05	0.07	0.03	0.07
CIM	0.04	0.11	0.03	0.04	0.02	0.05
ComPath	0.02	0.05	0.01	0.01	0.01	0.02
PaS	0.02	0.03	0.06	0.01	0.01	0.03

Table 8. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for WS-1000

	5%	**10%**	**15%**	**20%**	**25%**	**Average**
Rand	0.46	0.31	0.15	0.18	0.17	0.25
LDAG	0.09	0.11	0.08	0.02	0.06	0.07
MC-CELF	0.13	0.15	0.10	0.10	0.07	0.11
CEF++	0.09	0.11	0.05	0.07	0.04	0.07
CIM	0.06	0.09	0.06	0.00	0.02	0.05
ComPath	0.06	0.07	0.01	0.01	0.01	0.03
PaS	0.06	0.05	0.03	0.01	0.01	0.03

Table 9. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for LFR-5000

	5%	**10%**	**15%**	**20%**	**25%**	**Average**
Rand	3.75	3.75	3.50	2.50	2.20	3.14
LDAG	1.16	1.12	1.08	0.63	0.49	0.90
MC-CELF	1.38	1.38	1.16	0.70	0.52	1.03
CEF++	1.16	1.12	1.08	0.63	0.49	0.90
CIM	0.01	0.36	0.50	0.03	0.13	0.21
ComPath	0.00	0.36	0.18	0.04	0.12	0.14
PaS	1.13	1.12	1.08	0.62	0.48	0.88

Table 10. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for NetHEPT

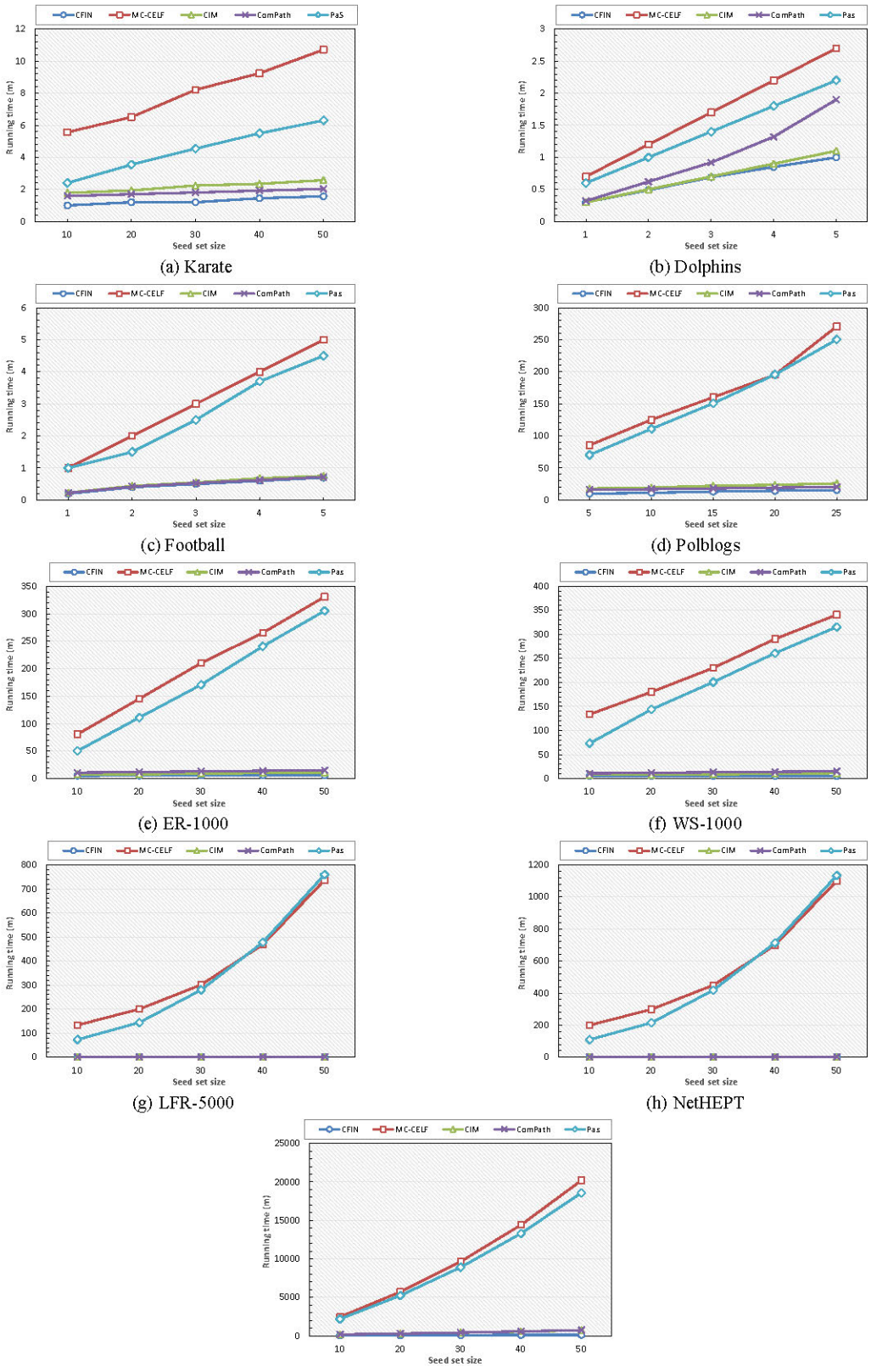
	5%	**10%**	**15%**	**20%**	**25%**	**Average**
Rand	0.31	2.05	2.75	2.04	1.09	1.65
LDAG	0.29	2.10	2.17	0.78	0.81	1.23
MC-CELF	0.01	0.43	0.78	0.72	0.75	0.54
CEF++	0.00	0.41	0.29	0.27	0.29	0.25
CIM	0.00	1.43	0.78	0.27	0.03	0.50
ComPath	0.00	1.39	0.78	0.27	0.03	0.49
PaS	0.01	1.39	0.78	0.72	0.73	0.72

Table 411. Comparison of the performance of the CFIN versus other algorithms in terms of the growth ratio for Epinions

	5%	**10%**	**15%**	**20%**	**25%**	**Average**
Rand	0.84	0.41	0.25	0.15	0.14	0.36
LDAG	0.04	0.03	0.06	0.06	0.06	0.05
MC-CELF	0.02	0.02	0.09	0.03	0.04	0.04
CEF++	0.01	0.02	0.03	0.02	0.03	0.02
CIM	0.01	0.02	0.02	0.01	0.01	0.02
ComPath	0.01	0.01	0.01	0.01	0.01	0.01
PaS	0.01	0.01	0.03	0.02	0.02	0.02

4.3.4. Experiment IV

In order to study the efficiency of the algorithm in terms of the running to reach the final influence spread, we compared the running time of the CFIN algorithm with MC-CELF, CIM, ComPath, and PaS. Since the size of the test network is different, we choose varying randomly selected seed set of nodes from 1, 2, 3, 4 to 5 for small size network including Karate, Dolphins, and football. For Political Blog, we use different sizes of seed set from 5, 10, 15, 20 to 25 nodes. For large size networks, we set different sizes of seed set from 10 to 50 nodes with 10 nodes interval. The results of running time as minutes for 10,000 times for MC simulations over the influence spread is depicted in figure 10.



(a) Karate (b) Dolphins

(c) Football (d) Polblogs

(e) ER-1000 (f) WS-1000

(g) LFR-5000 (h) NetHEPT

(i) Epinions

Figure 10: comparing running time of the algorithms for different sizes of seed set

From the result, we can see that the running time given by the CFIN is very close to the that of CIM and ComPath with near-linear behavior, this reason may be due to the effect of community finding. The results of figure 10 show that the speed of the CFIN is faster than that of the counterpart algorithms because it may first decide relevant

communities. Then the crucial nodes are selected from the current communities. Finally, CFIN uses local clustering co-efficient for spreading to bound the spreading within communities. Among the results of the algorithm in comparison with the CFIN, the CIM is slower because it picks relevant communities based on their volume and the relationship between them. The CIM also picks seeds faster since it immediately ignores a large portion of the network that has less effect.

Moreover, the CFIN tried to identify the initial nodes among the members of the selected nodes set. While MC-CELF carries out the calculations among all nodes of the input graph, this makes it time-consuming. The PaS algorithm is not slower than the CFIN, because PaS is not proper for social networks with various internal structures such as NetHEPT and Epinions datasets. For other algorithms, similar results can be obtained.

Next, to compare the running time consumed by the algorithms for choosing seed set nodes, we represent the corresponding *Dolan-Moré* performance profile. The *Dolan-Moré* performance profile was first introduced in [72] as an appropriate method to compare different algorithms in solving specific test problems based on proper criteria such as time, number of iterations, and number of activated nodes. To form a *Dolan-Moré* time profile for solving test problems $p \in P$ using different solvers $s \in S$, we first compute the ratio

$$r_{p,s} = \frac{t_{p,s}}{min\{t_{p,s}, s \in S\}} \tag{11}$$

where $t_{p,s}$ is the running time for solving the test problem p using solver s. To obtain an overall assessment of the performance of each solver, we compute

$$\rho_s(\tau) = \frac{1}{n_p}\{p \in P, r_{p,s} \leq \tau\} \tag{12}$$

which is, in fact, equal to the proportion of test problems solved by s in at most τ times the minimum running time among all solvers. Now, to sketch the *Dolan-Moré* profile, it is sufficient to plot $\rho_s(\tau)$ versus τ.

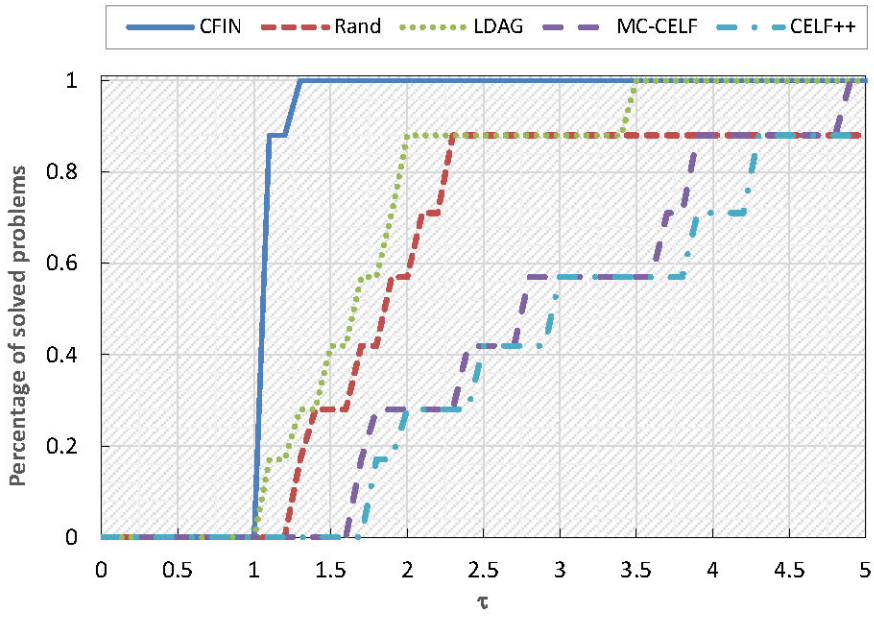


Figure 11. Comparison of the CFIN with other algorithms in terms of the *Dolan-Moré* performance profile

The represented *Dolan-Moré* performance profile in figure 11 confirms that the running time of the CFIN algorithm is less than that of the other algorithms. Although the obtained number of the final activated nodes in the network is similar in LDAG, CELF and MC-CELF, the CFIN algorithm reaches the solution significantly faster. We note that the greater running time for MC-CELF, and CELF++ can be affirmed based on the existence of the heavy Monte Carlo simulation as a time-consuming process. The LDAG algorithm is analogous with the CFIN algorithm in terms of running time, but it is not accurate in selecting high-quality initial seed.

5 Conclusion

In this paper, we investigated a novel algorithm called Community Finding Influential Nodes (CFIN) that selects k influential nodes based on community structures for influence maximization. The CFIN consists of two parts, including seed selection and local community spreading. The first part of the CFIN algorithm is the extraction of seed nodes from communities that uses the DLACD algorithm to detect communities from the input network. After community detection, in the second part of the CFIN, k communities based on their density and size are selected. Also, the CFIN algorithm selects k nodes with high degree centrality in terms of the initial nodes. Ultimately, the final seed nodes entered to distribute the local spreading by the modified simple path inside communities. Since nodes within communities have the same preferences, the CFIN investigated all the simple paths inside communities to achieve more affected nodes and cause a reasonable decrease in running-time. According to the experimental simulations, the obtained result of the algorithm showed that the CFIN outperforms other algorithms in terms of influence spread and the number of activated nodes at the end of the diffusion process. As the future works, we intend to extend the CFIN algorithm in multilayer social networks. People nowadays are a member of more than a social network, therefore studying these networks by considering overlapping properties may lead to more realistic and applicable results in the future.

Acknowledgement

This research was in part supported by a grant from IPM. (No. CS1398-4-222).

References

1. Alrashed S (2017) Reducing power consumption of non-preemptive real-time systems. J Supercomput 73:5402–5413. https://doi.org/10.1007/s11227-017-2092-9
2. Min-Allah N, Qureshi MB, Alrashed S, Rana OF (2019) Cost efficient resource allocation for real-time tasks in embedded systems. Sustain Cities Soc 48:101523. https://doi.org/10.1016/j.scs.2019.101523
3. Domingos P, Richardson M (2001) Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 57–66
4. Tang Y, Shi Y, Xiao X (2015) Influence maximization in near-linear time: A martingale approach. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp 1539–1554
5. Zhang H, Mishra S, Thai MT, et al (2014) Recent advances in information diffusion and influence maximization in complex social networks. Opportunistic Mob Soc Netw 37:37
6. Budak C, Agrawal D, El Abbadi A (2011) Limiting the spread of misinformation in social networks. In: Proceedings of the 20th international conference on World wide web. ACM, pp 665–674
7. Wu P, Pan L (2017) Scalable influence blocking maximization in social networks under competitive independent cascade models. Comput Netw 123:38–50
8. Feng Z, Xu X, Yuruk N, Schweiger TA (2007) A novel similarity-based modularity function for graph partitioning. In: International Conference on Data Warehousing and Knowledge Discovery. Springer, pp 385–396
9. Teng YW TC, Yu PS, Chen MS (2018) Revenue maximization on the multi-grade product. SIAM Int Conf Data Min 576–584
10. Ma H, Yang H, Lyu MR, King I (2008) Mining social networks using heat diffusion processes for marketing candidates selection. In: Proceedings of the 17th ACM conference on Information and knowledge management. ACM, pp 233–242
11. Khomami MMD, Rezvanian A, Bagherpour N, Meybodi MR (2018) Minimum positive influence dominating set and its application in influence maximization: a learning automata approach. Appl Intell 48:570–593
12. Rezvanian A, Moradabadi B, Ghavipour M, et al (2019) Social Influence Maximization. In: Rezvanian A, Moradabadi B, Ghavipour M, et al (eds) Learning Automata Approach for Social Networks. Springer International Publishing, Cham, pp 315–329
13. Kempe D, Kleinberg J, Tardos É (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. ACM, New York, NY, USA, pp 137–146
14. Leskovec J, Krause A, Guestrin C, et al (2007) Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 420–429
15. Goyal A, Lu W, Lakshmanan LV (2011) Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th international conference companion on World wide web. ACM, pp 47–48
16. Kundu S, Pal SK (2015) Deprecation based greedy strategy for target set selection in large scale social networks. Inf Sci 316:107–122
17. Zhou C, Zhang P, Zang W, Guo L (2015) On the upper bounds of spread for greedy algorithms in social network influence maximization. IEEE Trans Knowl Data Eng 27:2770–2783
18. Song G, Li Y, Chen X, et al (2016) Influential node tracking on dynamic social network: An interchange greedy approach. IEEE Trans Knowl Data Eng 29:359–372
19. Chen W, Wang Y, Yang S (2009) Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 199–208

20. Guo J, Zhang P, Zhou C, et al (2013) Item-based top-k influential user discovery in social networks. In: 2013 IEEE 13th International Conference on Data Mining Workshops. IEEE, pp 780–787
21. Zhao X-Y, Huang B, Tang M, et al (2015) Identifying effective multiple spreaders by coloring complex networks. EPL Europhys Lett 108:68005
22. Kim J, Kim S-K, Yu H (2013) Scalable and parallelizable processing of influence maximization for large-scale social networks? In: 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, pp 266–277
23. Kim J, Lee W, Yu H (2014) CT-IC: Continuously activated and time-restricted independent cascade model for viral marketing. Knowl-Based Syst 62:57–68
24. Li D, Xu Z-M, Chakraborty N, et al (2014) Polarity related influence maximization in signed social networks. PloS One 9:e102199
25. Luo Z-L, Cai W-D, Li Y-J, Peng D (2012) A pagerank-based heuristic algorithm for influence maximization in the social network. In: Recent progress in data engineering and internet technology. Springer, pp 485–490
26. Kimura M, Saito K, Nakano R, Motoda H (2010) Extracting influential nodes on a social network for information diffusion. Data Min Knowl Discov 20:70
27. Ohsaka N, Akiba T, Yoshida Y, Kawarabayashi K (2014) Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In: Twenty-Eighth AAAI Conference on Artificial Intelligence
28. Goyal A, Lu W, Lakshmanan LV (2011) Simpath: An efficient algorithm for influence maximization under the linear threshold model. In: Data Mining (ICDM), 2011 IEEE 11th International Conference on. IEEE, pp 211–220
29. Chen W, Yuan Y, Zhang L (2010) Scalable influence maximization in social networks under the linear threshold model. In: 2010 IEEE international conference on data mining. IEEE, pp 88–97
30. Lu Z, Fan L, Wu W, et al (2014) Efficient influence spread estimation for influence maximization under the linear threshold model. Comput Soc Netw 1:2
31. Heidari M, Asadpour M, Faili H (2015) SMG: Fast scalable greedy algorithm for influence maximization in social networks. Phys Stat Mech Its Appl 420:124–133
32. Narayanam R, Narahari Y (2011) A shapley value-based approach to discover influential nodes in social networks. IEEE Trans Autom Sci Eng 8:130–147
33. Cantwell GT, Newman MEJ (2019) Mixing patterns and individual differences in networks. Phys Rev E 99:042306
34. Riolo MA, Cantwell GT, Reinert G, Newman ME (2017) Efficient method for estimating the number of communities in a network. Phys Rev E 96:032310
35. Liu W, Pellegrini M, Wang X (2014) Detecting communities based on network topology. Sci Rep 4:5739
36. Li H, Bhowmick SS, Sun A, Cui J (2015) Conformity-aware influence maximization in online social networks. VLDB J 24:117–141
37. Guo L, Zhang D, Cong G, et al (2016) Influence maximization in trajectory databases. IEEE Trans Knowl Data Eng 29:627–641
38. Li Y, Zhang D, Tan K-L (2015) Real-time targeted influence maximization for online advertisements
39. Stein S, Eshghi S, Maghsudi S, et al (2017) Heuristic Algorithms for Influence Maximization in Partially Observable Social Networks. In: SocInf@ IJCAI. pp 20–32
40. Wilder B, Immorlica N, Rice E, Tambe M (2018) Maximizing influence in an unknown social network. In: Thirty-Second AAAI Conference on Artificial Intelligence
41. Rezvanian A, Moradabadi B, Ghavipour M, et al (2019) Social Community Detection. In: Learning Automata Approach for Social Networks. Springer International Publishing, Cham, pp 151–168
42. de Guzzi Bagnato G, Ronqui JRF, Travieso G (2018) Community detection in networks using self-avoiding random walks. Phys Stat Mech Its Appl 505:1046–1055
43. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
44. Malliaros FD, Vazirgiannis M (2013) Clustering and community detection in directed networks: A survey. Phys Rep 533:95–142. https://doi.org/10.1016/j.physrep.2013.08.002
45. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76:036106
46. Gregory S (2010) Finding overlapping communities in networks by label propagation. New J Phys 12:103018
47. Xie J, Szymanski BK (2012) Towards linear time overlapping community detection in social networks. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, pp 25–36
48. Ugander J, Backstrom L (2013) Balanced label propagation for partitioning massive graphs. In: Proceedings of the sixth ACM international conference on Web search and data mining. pp 507–516
49. Stokes ME, Barmada MM, Kamboh MI, Visweswaran S (2014) The application of network label propagation to rank biomarkers in genome-wide Alzheimer's data. BMC Genomics 15:282
50. Hosseini R, Rezvanian A (2020) AntLP: ant-based label propagation algorithm for community detection in social networks. CAAI Trans Intell Technol 5:34–41
51. Kuzmin K, Shah SY, Szymanski BK (2013) Parallel overlapping community detection with SLPA. In: 2013 International Conference on Social Computing. IEEE, pp 204–212
52. Ahn Y-Y, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. nature 466:761–764
53. Ye Q, Wu B, Zhao Z, Wang B (2011) Detecting link communities in massive networks. In: 2011 International Conference on Advances in Social Networks Analysis and Mining. IEEE, pp 71–78
54. Lee C, Reid F, McDaid A, Hurley N (2010) Detecting highly overlapping community structure by greedy clique expansion. ArXiv Prepr ArXiv10021827

55. Zhang X, Wang C, Su Y, et al (2017) A fast overlapping community detection algorithm based on weak cliques for large-scale networks. IEEE Trans Comput Soc Syst 4:218–230
56. Badie R, Aleahmad A, Asadpour M, Rahgozar M (2013) An efficient agent-based algorithm for overlapping community detection using nodes' closeness. Phys Stat Mech Its Appl 392:5231–5247
57. Khomami MMD, Rezvanian A, Meybodi MR (2016) Distributed learning automata-based algorithm for community detection in complex networks. Int J Mod Phys B 30:1650042
58. Girvan M, Newman ME (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99:7821–7826
59. Lusseau D (2003) The emergent properties of a dolphin social network. Proc R Soc Lond B Biol Sci 270:S186–S188
60. Park J, Newman ME (2005) A network-based ranking system for US college football. J Stat Mech Theory Exp 2005:P10014
61. Adamic LA, Glance N (2005) The political blogosphere and the 2004 US election: divided they blog. In: Proceedings of the 3rd international workshop on Link discovery. ACM, pp 36–43
62. Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: Densification and shrinking diameters. ACM Trans Knowl Discov Data TKDD 1:1–41
63. Richardson M, Agrawal R, Domingos P (2003) Trust management for the semantic web. In: International semantic Web conference. Springer, pp 351–368
64. Erdos P, Rényi A (1960) On the evolution of random graphs. Publ Math Inst Hung Acad Sci 5:17–61
65. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world' networks. Nature 393:440–442
66. Lancichinetti A, Radicchi F, Ramasco JJ, Fortunato S (2011) Finding statistically significant communities in networks. PloS One 6:e18961
67. Goyal A, Bonchi F, Lakshmanan LVS (2011) A data-based approach to social influence maximization. Proc VLDB Endow 5:73–84
68. Chen Y-C, Zhu W-Y, Peng W-C, et al (2014) CIM: Community-based influence maximization in social networks. ACM Trans Intell Syst Technol TIST 5:25
69. Rahimkhani K, Aleahmad A, Rahgozar M, Moeini A (2015) A fast algorithm for finding most influential people based on the linear threshold model. Expert Syst Appl 42:1353–1361
70. Ok J, Jin Y, Shin J, Yi Y (2014) On maximizing diffusion speed in social networks: impact of random seeding and clustering. In: The 2014 ACM international conference on Measurement and modeling of computer systems. pp 301–313
71. He J-L, Fu Y, Chen D-B (2015) A novel top-k strategy for influence maximization in complex networks with community structure. PloS One 10:
72. Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91:201–213