Minimum Positive Influence Dominating Set and its Application in Influence Maximization: a Learning Automata Approach

Mohammad Mehdi Daliri Khomami[1], Alireza Rezvanian[2,3], Negin Bagherpour[4] and Mohammad Reza Meybodi[2]

[1] Computer Engineering and Information Technology Department, Islamic Azad University, Qazvin branch, Qazvin, Iran

[2] Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

[3] Department of Computer Engineering and Information Technology, Hamedan University of Technology, Hamedan, Iran

[4] Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran

Abstract

In recent years, with the rapid development of online social networks, an enormous amount of information has been generated and diffused by human interactions through online social networks. The availability of information diffused by users of online social networks has facilitated the investigation of information diffusion and influence maximization. In this paper, we focus on the influence maximization problem in social networks, which refers to the identification of a small subset of target nodes for maximizing the spread of influence under a given diffusion model. We first propose a learning automaton-based algorithm for solving the minimum positive influence dominating set (MPIDS) problem, and then use the MPIDS for influence maximization in online social networks. We also prove that by proper choice of the parameters of the algorithm, the probability of finding the MPIDS can be made as close to unity as possible. Experimental simulations on real and synthetic networks confirm the superiority of the algorithm for finding the MPIDS. Experimental results also show that finding initial target seeds for influence maximization using the MPIDS outperforms well-known existing algorithms.

Keywords. Learning Automata, Social Network Analysis, Minimum Positive Influence Dominating Set, Influence Maximization.

1 Introduction

Today, online social networks (OSNs) such as *Facebook*, *LinkedIn* and *Twitter* play an important role in diffusing opinions and information between individuals. Extensive research on social network analysis shows that online users, when sharing information, trust the opinions or information obtained from their close social groups such as friends and acquaintances far more

than they trust the comments of ordinary people or news about information from public media. Moreover, with the rapid development and increasing popularity of online social networks, an enormous amount of information has been generated and diffused by human interactions through these networks. The availability of information diffused by online users had led to emerging opportunities for enabling large-scale research into information diffusion.

Extensive studies have been conducted on many real and complex social networks. In most of the studies, a social network can usually be modeled as a graph, with a set of nodes representing online users and edges representing a relationship between users. Social networks have the small-world property, implying a short average distance between each pair of nodes in the real network [1]. Another significant characteristic of social networks is the community structure, that is, the tendency to form communities between some nodes [2]. Furthermore, social networks reflect the scale-free property, in which several nodes with a relatively large number of edges exist that these nodes are known as hubs [3].

One of the fundamental problems in information diffusion in social networks is the influence maximization problem. The problem of optimizing influence maximization in a social network focuses on how to select a small sub-set of individuals as the initial influence adopters to trigger a cascade such that the influence diffusion in the social network is maximized. This problem has drawn a great deal of attention because it can be applied to optimize the spread of information, new ideas or innovations through social networks. One of the most popular applications is viral marketing. For a company needing to promote a new service, viral marketing is an effective strategy for a limited budget. Instead of promoting the new service to all potential customers, a company can target a small sub-set of customers who then recommend the service to their friends or acquaintances. Due to the word-of-mouth effect, a small number of nodes can affect the whole network, while the idea, service or innovation can be spread widely [4]. This set of nodes is referred to as a dominating set (DS). Because there is a tendency in online social networks for users to match each other's behaviors, the extended minimum dominating set could be investigated using the DS concept as a positive influence dominating set (PIDS) in social networks.

Let $G=\langle V, E\rangle$ be the graph of a network, in which V is the set of nodes and E is the set of edges. For such a network, a PIDS is a subset $P \subset V$ with a minimum size such that any node $v_i \in V$ is dominated by at least $\left\lceil \frac{n_i}{2} \right\rceil$ nodes in P, where n_i is the degree of node v_i. The minimum positive influence dominating set (MPIDS) is the PIDS with minimum cardinality [5]. The MPIDS concept has many applications in online social networks. For example, in terms of viral marketing [6], an influential individual can efficiently change the decisions of other people to adopt or refuse a product or service. Due to structural changes in positive effects on social networks, it can be guaranteed through the constructing of an MPIDS that each node has more positive neighbors than it has negative ones. Figure 1 provides a typical example of an MPIDS. The black nodes, P= {3, 4, 5, 6}, represent one of the MPIDSs in this network. We note that {1, 3, 5, 6} and {2, 3, 5, 6} are two other possible choices.

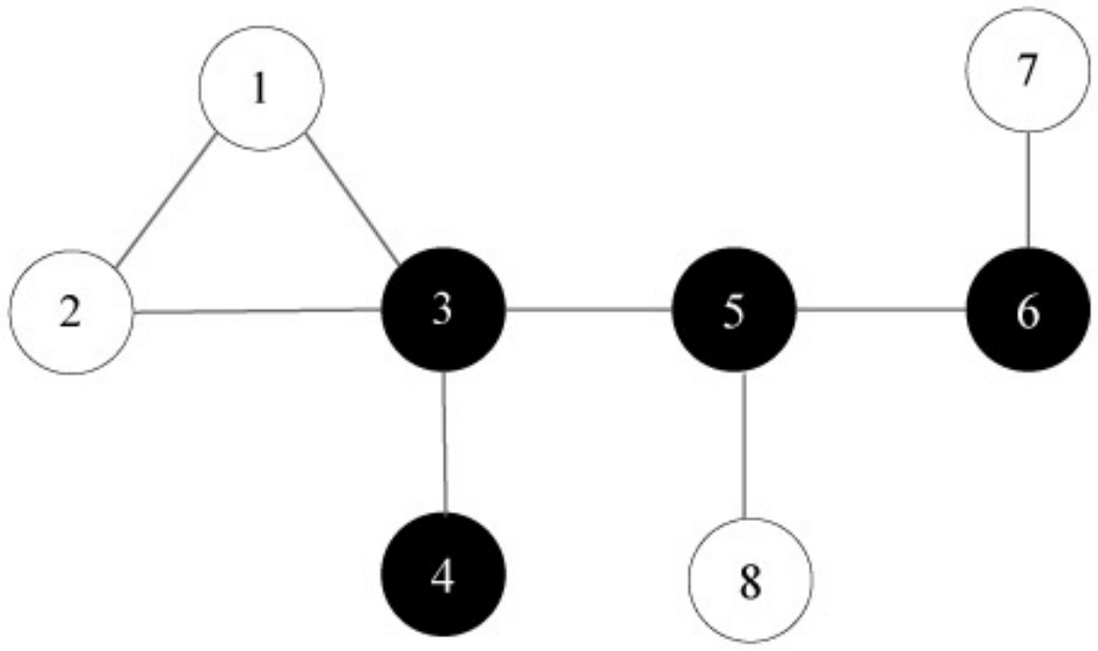


Figure 1. Example of a minimum positive influence dominating set in a typical input network.

Several studies have been conducted on finding PIDS; these studies can be summarized as follows. *Wang et al.* [6] proposed a greedy algorithm for finding the MPIDS in a social network with computational time complexity $O(n^3)$. The proposed algorithm uses classic dominating sets iteratively and adds nodes with an $H(d)$, where H is the harmonic function and d is the maximum vertex degree of the network. Moreover, the authors prove that the problem is APX-hard. *Raei et al.* [7] presented another greedy algorithm for the PIDS problem, in which the running time is reduced to $O(n^2)$. Although the greedy algorithm was successful in finding the optimal solution, it failed to do so in some cases and obtained an inappropriate solution. Another greedy algorithm which has been designed for power law graphs was introduced by Zhang et al. [8]. This work also showed that the presented algorithm has a constant approximation factor for power law graphs; however, the authors did not consider real networks, which do not necessarily reflect power-law degree distribution properties. *Dinh et al.* [9] defined the new concept of the total positive influence dominating set (TPIDS), in which the goal is to cover the fraction ρ of neighbors rather than $\left\lceil \frac{n_i}{2} \right\rceil$. The authors also presented an approximation factor for the TPIDS and showed that in power-law networks, in which the degree sequence follows a power-law distribution, both PIDS and TPIDS problems admit constant factor approximation algorithms. Furthermore, they investigated this problem only in the special case in which the influence spread is performed based on a linear threshold model. In their model, a node is activated when a fraction ρ of neighbors are activated. *Wang et al.* [10] considered the slightly different problem of the weighted positive influence dominating set. In this problem, each node of the network is assumed to have a weight value, and the goal is to identify the PIDS with maximum total weight.

A significant application of the PIDS is the solution of the influence maximization problem. Given a social network, the problem of influence maximization involves the determination of a set of nodes that maximizes the spread of influences based on the given propagation model. In [4], two classical propagation models taken from mathematical sociology are presented, including the linear threshold model and the independent cascade model. In the linear threshold (LT) model, each node has a threshold that is chosen uniformly and randomly between 0 and 1. A node is activated when the fraction of its activated neighbors reaches this threshold. Threshold models have been proposed and studied in many domains, for example, sociology, as a reasonable approach toward a model for influence in a social network. In the independent cascade (IC) model, each node has either an active or an inactive state and is allowed to change its state from inactive

to active, but not vice versa. Each active node has a single and independent chance to activate its inactive neighbors. *Java et al.* [11] applied the linear threshold model for finding influential bloggers to maximize the spread of information throughout the blogosphere. They also investigated the performance of the different algorithms, such as PageRank, HITS and in-degree, on modeling the influence of blogs. In [12], *Kimura and Saito* used the shortest path model for information diffusion and proposed an algorithm to achieve the maximum spread under their proposed model. They investigated an algorithm that efficiently computes a proper approximate solution to the problem using a greedy approach. *Domingos and Richardson* [13, 14] considered influence maximization a basic algorithmic problem and suggested a probabilistic model of interaction between people in networks. Following this idea, *Kempe et al.* [4] formulated influence maximization as a discrete optimization problem that attempts to find a set of *k* nodes according to a specific propagation model to maximize the influence spread.

In [15], a new algorithm was presented by *Liu et al.* for solving the influence maximization problem by forming an influence-spreading path. They argued that time plays an important role in the influence spread between two users and that the time needed for each user to influence another is different. Another algorithm is investigated by *Xu et al*. [16]; the authors solved the problem of influence maximization using a weighted maximum cut problem and a semi-definite program-based algorithm. Based on the proposed algorithm, the value of the influence between two non-adjacent individuals is computed according to the existence of a social influence path between them. In addition, *Jong et al.* [17] presented an algorithm for influence maximization on specific users in social networks and formulated the problem as query processing to find specific users within the network as a whole. One of the most important heuristic algorithms is the high-degree approach suggested in [4]. According to this algorithm, the initial *k* nodes used as seeds are nodes with *k* maximum output degree. In the *PageRank* algorithm for influence maximization [18], nodes are ranked according to significance, although recent research has shown that *PageRank* yields inappropriate results for finding influential nodes. In the hyperlink-induced topic search (HITS) [19], the concepts of hub and authority were investigated, in which a hub is a page that points to many other pages, and a good authority represents a page that is linked to by many different hubs. *Lu* et al. [20] presented the extension of the *H*-index concept to determine how important a node is in the underlying network as an important parameter to quantify the influential spreaders. Based on the definition, the H-index of a node is specified to be the maximum value *h* such that there exists at least *h* neighbors of degree no less than *h* in a network. They also discovered that the *H*-index of a node can be a better predictor of the influence of a node in epidemic spreading than the degree or the coreness. A Cost-Effective Lazy Forward selection (CELF) algorithm [21], is a well-known lazy-forward greedy algorithm with an optimization, in which new seed nodes are selected by further exploiting the sub-modular property of influence maximization. More precisely, the marginal gain of a node in the current iteration cannot be greater than its marginal gain in previous iterations; hence, the number of spread estimation calls can be reduced significantly.
To improve the effectiveness and performance of CELF, CELF++ [22] is presented as an extension of CELF. In CELF++, the influence spread for two successive iterations of the greedy algorithm is computed simultaneously, leading to a lower number of iterations being required. More recently, *Zeng et al.* [1] proposed the Mixed Degree Decomposition (MDD) algorithm, which cooperates with both the residual degree (number of links connecting to the remaining nodes) and exhausted

degree (number of links connecting to the removed nodes) in networks and is therefore called mixed degree. During the execution of the MDD algorithm, nodes are removed based on the mixed degree. In *Yeruva et al.* [2] a *Pareto*-Shell decomposition algorithm is proposed such that the algorithm selects non-dominated spreads with a ratio of high out-degree to in-degree and high in-degree. In addition, they have shown experimentally that the proposed algorithm outperforms high degree centrality in some cases.

A valuable direction for research into finding influential nodes in graph theory is the solution of the minimum dominating set (MDS) problem. Several algorithms have been proposed [23–28] to solve this problem. Because finding the MDS and its variants are NP-complete problems [23], the algorithms presented in [23–28] attempt to minimize the cardinality of the obtained dominating set in only certain areas. In [29–31], the most popular approximations based on greedy algorithms are presented. In greedy selection, the main idea is to choose a node with maximum dominance among uncovered nodes until a set with minimum cardinality is achieved. This idea results in an $Ln(\Delta)$ approximation factor, where Δ denotes the maximum degree in the graph. A polynomial time approximation scheme (PTAS) for polynomial bounded graphs is presented in [32] to solve the MDS problem in polynomial bounded graphs. A polynomial bounded graph is a graph in which the number of independent nodes in the r-hop neighborhood of a node is bounded by a polynomial function. In [33], the MDS is considered in terms of critical nodes in social networks, while it is shown, as a significant property, that in scale-free networks a high-degree node tends to be included in the dominating set. It is also shown experimentally that the obtained MDS is unique when the average degree is large.

In this paper, we propose an algorithm based on learning automata for finding the near-optimal MPIDS in social networks. We note that for a lower cardinality of the obtained PIDS, influence will be spread more efficiently because a smaller number of nodes is involved in the spread process. In the proposed algorithm, a learning automaton is assigned to each node of the network. During the execution of the proposed algorithm, the set of learning automata cooperate with each other to find the near-optimal positive influence dominating set in the given network. To investigate the performance of the proposed algorithm, several experiments have been conducted on well-known synthetic and real datasets.

The rest of the paper is organized as follows. Section 2 provides a necessary background for an introduction to learning automata, entropy and influence maximization. Section 3 presents the proposed learning automaton-based algorithm for finding the minimum positive influence dominating set in social networks. In section 4, the convergence of our proposed algorithm to a PIDS is proved, and numerical results are reported in section 5. Finally, section 6 presents our concluding remarks.

2 Background

2.1 Learning Automata

A learning automaton (LA) [34] is an adaptive decision-making unit in which the performance is improved by learning how to choose the optimal action from a finite set of allowed actions considering repeated interactions with a random environment. The action is chosen randomly

based on a probability distribution that is retained over the action set so that at each instant the given action serves as the input to the random environment. In turn, the environment responds to the action taken with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action set so that the average penalty received from the environment is minimized. The environment can be described by a triple $E=\{\alpha, \beta, c\}$, where $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of the inputs, $\beta=\{\beta_1, \beta_2, \ldots, \beta_m\}$ denotes the set of values which are chosen from the reinforcement signal and $c=\{c_1, c_2, \ldots, c_m\}$ denotes the set of penalty probabilities, where c_i is associated with the given action α_i. If the penalty probabilities are constant, the random environment is said to be a stationary random environment; otherwise, it is known as a non-stationary environment. The environments, depending upon the nature of the reinforcement signal β, can be classified into P-model, Q-model and S-model. If the reinforcement signal is only able to take two binary values 0 and 1, the corresponding environment is referred to as a P-model environment. Another type of environment that allows a finite number of values in the interval [0, 1] to be taken by the reinforcement signal is known as a Q-model environment. In S-model environments, however, the reinforcement signal lies in the interval [0, 1]. The relationship between the learning automaton and its random environment is shown in Figure 2.

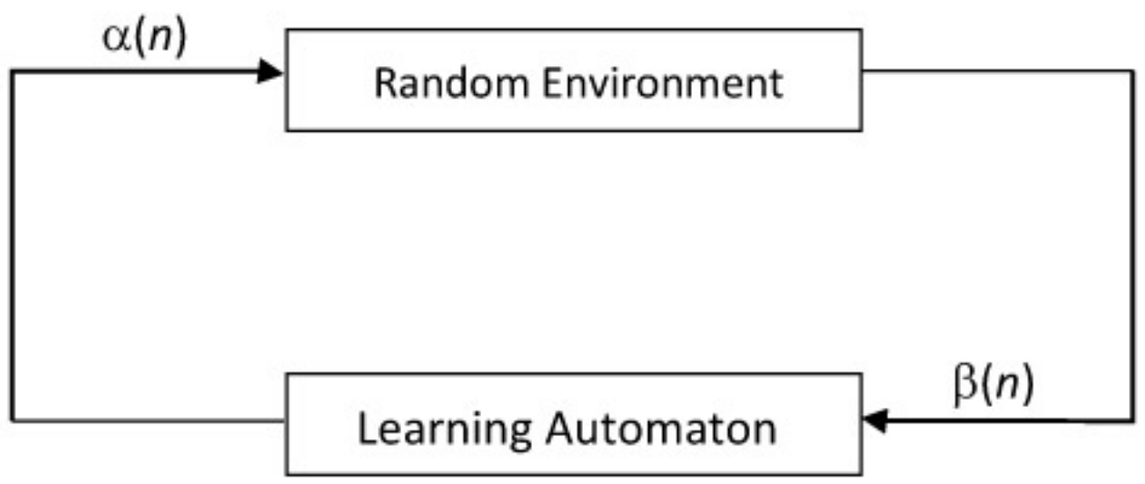


Figure 2. The relationship between the learning automaton and its random environment.

Learning automata can be also classified into two main groups: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $\langle \beta, \alpha, T \rangle$, in which β is the set of inputs, α is the set of actions and T is the learning algorithm. The learning algorithm is a recurrence relation that is used to modify the action probability vector. Let $\alpha(k)$ and $p(k)$ be the action chosen at instant k and the action probability vector on which the chosen action is based, respectively. The recurrence equations (1) and (2) introduce a linear learning algorithm by which the action probability vector p is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k. Then, to reward (i.e., $\beta(k)=0$) and penalize (i.e., $\beta(k)=1$) an action, the relationships

$$p_j(k+1) = \begin{cases} p_j(k) + a(k) \times [1 - p_j(k)] & if\ i = j \\ p_j(k) - a(k) \times p_j(k) & if\ i \neq j \end{cases} \quad (1)$$

and

$$p_j(k+1) = \begin{cases} p_j(k) \times (1 - b(k)) & if\ i = j \\ \dfrac{b}{r-1} + p_j(k)(1 - b(k)) & if\ i \neq j \end{cases} \quad (2)$$

are used respectively, where *r* is the number of actions that can be chosen by the automaton and $a(k)$ and $b(k)$ denote the reward and penalty parameters determining the amount of increase and decrease in action probabilities, respectively. When $a(k)$ equals $b(k)$, the recurrence equations (1) and (2) are referred to collectively as the linear reward penalty (L_{R-P}) algorithm. If $a(k) \gg b(k)$, the given equations are known as the linear reward-ε penalty ($L_{R-\varepsilon P}$). Finally if $b(k) = 0$, they are known as the linear reward-inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged during the penalizing process.

Leaning automata have been successfully used in many applications such as image processing [35, 36], combinatorial optimization problems [37–39], stochastic graphs [37, 40–42], wireless networks [43], social networks [44–46], network sampling [47], grid computing [48], cloud computing [49], and adaptive petri-net [50], to mention only a few.

2.2 Entropy

Entropy is introduced into reinforcement learning as a measure of the learning process [51]. Entropy is a fundamental concept in thermodynamics, representing the degree of order or disorder in a thermodynamic system, and plays an important role in various fields of computer science such as learning. Entropy essentially estimates the measure of uncertainty in the action of a learning automaton during the next selection. A larger value of entropy leading to more uncertainty in the probability vector for a learning automaton means that the automaton contains no useful information for achieving a goal and its performance is more or less similar to a random selection. However, lower entropy indicates that by decreasing the uncertainty measure the automaton selects one of the actions with a higher likelihood, which is useful information in moving closer to the target.

Suppose that $\{p_1, p_2, \ldots, p_r\}$ is the action probability vector of a learning automaton with *r* actions and that the entropy of the action probability vector is determined by

$$E(P) = -\sum_{i=1}^{r} p_i \quad \log(p_i). \quad (3)$$

The entropy value is maximized when the probabilities of all actions are the same. Over successive iterations, the entropy value will decrease [51]. The learning automaton initially assigns equal probabilities to all actions. This uniform distribution leads to considerable uncertainty. Hence, by using the entropy of the action probability vector as the convergence criterion, the learning automaton converges to an optimal action with a probability close to unity in the learning process. In the case of a two-action learning automaton, the entropy will have a value between 0 and 1. Otherwise, the entropy value can be re-scaled to the interval [0, 1].

2.3 Influence Maximization

Influence maximization is one of the most important issues in the field of social networks, the main idea of which is taken from viral marketing. The point of interest in viral marketing is how an innovation is spread throughout a population through word of mouth to maximize the adaptation in the whole network. In influence maximization, the goal is to find a set of nodes with minimum cardinality such that their activation maximizes the influence spread through the network. To this end, we first find the MPIDS in the network using Algorithm 1 (i.e., figure 3). To obtain proper candidate nodes for influence spread, we select *k* nodes from the obtained MPIDS based on an appropriate criterion that can be set according to the application to random, degree, betweenness [52] or closeness [53]. With the aid of the LTM model [4], we next spread influence from these *k* nodes.

One of the most important criteria for the selection of an initial active node is betweenness, which is calculated for a specific node by computing the number of shortest paths going through a node. Betweenness is formulated for node *v* as

$$Betweenness = \sum_{i \neq j, i \neq v, j \neq v} g_{ivj}/g_{ij} \tag{4}$$

where g_{ivj} indicates the total length of the shortest path from node *i* to node *j* through node *v*, and g_{ij} shows the total length of the shortest path from node *i* to node *j* in the network.

The closeness centrality of a node is defined by the inverse of the average length of the shortest path from all other nodes in the networks and is defined as

$$Closeness = \frac{1}{\sum_{i \neq v} g_{i,v}}. \tag{5}$$

The eccentricity for a specific node in a connected network is the maximum length of the shortest path from the other nodes in the network. The maximum eccentricity shows the diameter of the network.

3 Proposed Algorithm for Finding the Positive Influence Dominating Set

In this section, we propose a learning automaton-based algorithm for solving the minimum positive influence dominating set problem. The undirected graph $G=\langle V, E\rangle$ serves as the input to the algorithm and the desired output is the near-optimal minimum positive influence dominating set. The proposed learning automaton-based algorithm involves four steps. Following the initialization step by assigning a learning automaton to the nodes of the input network, the second step of the proposed algorithm tries to find a candidate positive influence dominating set with the help of learning automata. The algorithm then evaluates the cardinality of the obtained set to find the minimum positive influence dominating set. Finally, the probability vector is updated until the stopping criteria are satisfied. We describe the four steps of the proposed algorithm in detail in the following subsections.

3.1 Initialization

In the proposed algorithm, each node of the graph is initially equipped with a learning automaton. Thus, a network of learning automata isomorphic to the graph is initially constructed. The set of learning automata (LAs) can be described by a tuple $LA=\{A, \alpha\}$ where $A=\{A_1,\ldots,A_n\}$ corresponds to the learning automata and $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the finite set of actions. Moreover, a probability vector assigns a probability value to the selection of each action. A common choice for the initial value of the probability vector is computed using a uniform probability distribution. In Algorithm 1, the action set of each learning automaton A_i residing in node v_i consists of only two actions, α_i^0 and α_i^1. Let Ω_t be the selected positive influence dominating set in iteration t. If α_i^0 is selected, the corresponding node v_i becomes a member of Ω_t. Otherwise, if α_i^1 is selected, no change will occur in the membership of the positive influence dominating set.

3.2 Finding a Candidate Positive Influence Dominating Set

At each iteration, all learning automata simultaneously select one of their actions according to the action probability vector. In the proposed algorithm, the action α_i^0 corresponds to choosing v_i as a member of the currently obtained PIDS, while selecting α_i^1 means that v_i does not belong to the current PIDS. Hence, all learning automata that selected action α_i^0 become members of Ω_t. Let n_i be the number of nodes that are neighbors of node v_i; then the algorithm checks whether the obtained Ω_t is a PIDS by considering whether each node in the network has $\left\lceil \frac{n_i}{2} \right\rceil$ neighbors included in the current PIDS. If not, the process of action selection by the learning automata continues; otherwise, step 3.3 is started. We note that the process of action selection is performed simultaneously by all learning automata; this cooperation between learning automata helps to accelerate the learning process.

3.3 Computing the Objective Function

A dynamic threshold is applied by the algorithm to evaluate whether the current obtained solution is a proper one. Let the dynamic threshold be defined as the average cardinality of the positive influence dominating sets up to iteration t:

$$D_t = \frac{1}{t}\sum_{i=1}^{t} |\Omega_i|, \tag{6}$$

where Ω is the positive influence dominating set and $|\Omega_t|$ is the cardinality of the selected positive influence dominating set at iteration t. The dynamic threshold aids in deciding whether the obtained PIDS is a proper one; more precisely, if the dynamic threshold, D_t, is less than $|\Omega_t|$, the current PIDS is an appropriate candidate MPIDS.

3.4 Updating the Probability Vector

At every iteration t, if the average cardinality of the positive influence dominating set is less than or equal to the dynamic threshold D_t, then depending upon the internal state, the actions chosen by all learning automata are rewarded. Each learning automaton updates its action probability vector by using an L_{R-I} reinforcement scheme. Finding the positive influence dominating set and updating the action probabilities are continued until the entropy of the probability vector of the

chosen nodes reaches a certain threshold, or the number of resulting positive influence dominating sets reaches a pre-defined threshold. Figure 3 shows the steps of Algorithm 1 in detail.

Algorithm 1: proposed algorithm for finding the minimum positive influence dominating set

Input The graph $G=(V, E)$, Threshold P, T
Output: Minimum positive influence dominating set
Assumptions
Assign an automaton A_i to each node v_i, and the initial action set of each automaton is set equally;
Begin
Let t be the iteration number of the algorithm, initially set to 1;
Let Ω_t be the positive influence dominating set at iteration t, initially set to empty;
Let N_t be the set of non-positive influence dominating sets at iteration t, initially set to V;
Let D_t be the average size of all positive influence dominating sets up to iteration t;
Let P_t be the entropy value of probability of the nodes of the positive influence dominating set, initially set to 1;
While ($t< T$ or $P_t< P$)
Repeat
For all of the learning automata Do
Learning automaton A_i chooses an action using its action probability vector;
If (the action selected by A_i is a member of the positive influence dominating set) then // α^0_i
$\Omega_t \leftarrow \Omega_t \cup v_i$;
End
End for
Until (Ω_t construct a candidate positive influence dominating set)
Compute D_t according to Equation (7);
If ($|\Omega_t| \leq D_t$)) then
Reward the actions chosen by all of the activated learning automata;
Else
Penalize the actions chosen by all of the activated learning automata;
End If
Compute entropy probability P_t according to Equation (3);
$t \leftarrow t + 1$;
End while
End Algorithm

Figure 3. Pseudo-code for the proposed algorithm for finding the minimum positive influence dominating set.

The proposed positive influence dominating set is crucial in the application of influence maximization and can be used as prior knowledge to select the initial active node. Indeed, in order to spread diffusion throughout the network we select initial nodes from the obtained PIDS.

Note 3.1. In Algorithm 1 a penalizing parameter is assumed so that when penalizing an action is necessary, the probabilities are updated based on Equation (2). However, in our experiments the learning scheme L_{R-I} is used, in which the penalizing parameter is assumed equal to zero; hence, the actions are rewarded or the probabilities remain unchanged.

4 Convergence Results

In this section, we prove the convergence of Algorithm 1 to the optimal solution by a proper choice of learning rate.

Theorem 1. Let $q(t)$ be the probability of constructing a positive influence dominating set in a given graph at stage t. If $q(t)$ is updated according to Algorithm 1, then for every $\varepsilon > 0$, there exists a learning rate $\tilde{a} \in (0,1)$ such that for all $a \in (0, \tilde{a})$, we have

$$\text{Prob}\left(\lim_{t\to\infty} q(t)=1\right)\geq 1-\varepsilon. \tag{7}$$

Proof.

The sketch of the proof is similar to the method given in [54] and [34] to analyze of the behavior of the learning automaton operating in a non-stationary environment. The steps of the convergence proof are briefly outlined as follows. First, the convergence of the penalty probability of the PIDS to the final penalty probability is established for sufficiently large values of *t*. Moreover, it is shown that the probability of choosing the minimum PIDS is a sub-Martingale process for large values of *t*; hence, the change in the probability of constructing the minimum PIDS is non-negative at each step. Finally, the convergence of Algorithm 1 to the minimum PIDS is proved using well-known Martingale convergence results. For details and proofs of statements see e.g., [28].

Step 1. Assuming the generated PIDS at stage *t* is penalized with probability $c(t)$ and $\lim_{t\to\infty} c(t)=c^*$, for each $\varepsilon\in(0,1)$ and $t>T(\varepsilon)$ we have

$$prob\left[\left|c^*-c(t)\right|>0\right]<\varepsilon \tag{8}$$

where $T(\varepsilon)$ denotes the minimum number of stages of Algorithm 1 to achieve error rate ε.

Step 2. Let $c(t)=prob\left[|\Omega_t|>D_t\right]$ and $d(t)=1-c(t)$ be the probabilities of penalizing and rewarding the PIDS respectively. If $q(t)$ is updated according to Algorithm 1, then the conditional expectation of $q(t)$ is defined as $E\left[q(t+1)|q(t)\right]=q(t)\left[c(t)\,q(t)+d(t)\left(\prod_{v_m\notin P}\left(1-p_m^1(t)\right)\prod_{v_m\in P}p_m^1(t)\right)\right]$, where

$$p_m^1(t+1)=\begin{cases}p_m^1(t)+a\left(1-p_m^1(t)\right) & v_m\in P\\ p_m^1(t)(1-a) & v_m\in P\end{cases}, \tag{9}$$

and $p_m^1(t)$ is the probability of declaring v_m as a dominator by itself at stage *t*. Moreover, $\Delta q(t)=E\left[q(t+1)|q(t)\right]-q(t)$ is always non-negative.

Step 3. In defining $\Gamma(q)=prob\left[q(\infty)=1|q(\cdot)=q\right]$, the goal is to compute a lower bound for $\Gamma(q)$ that guarantees the convergence of Algorithm 1 after a sufficient number of stages. Let $\psi(q)$ be a sub-regular function on $[0,1]$ satisfying $\psi(1)=0$. We note that $\psi(q)\leq\Gamma(q)$ holds for any $q\in[0,1]$ (see, e.g., [27]). Hence, to establish the convergence of Algorithm 1 with probability equal to 1, it is sufficient to find a sub-regular function $\psi(q)$ that is bounded from below and converges to 1.

Let $\varphi(x,q)=\frac{e^{\frac{-xq}{a}}-1}{e^{\frac{-x}{a}}-1}$. In [34], it is shown that the sufficient condition for the sub-regularity of $\varphi(x,q)$ on $[0,1]$ is that

$$G(x,q)=(1-q)d^{*}V[-x(1-q)]-qd^{*}V[xq]\geq 0, \tag{10}$$

where $V[u]=\begin{cases}\frac{e^{u}-1}{u} & u\neq 0\\ 1 & u=0\end{cases}$. We note that equation (10) is equivalent to

$$\frac{V[-x(1-q)]}{V[xq]}\geq\frac{q}{1-q}. \tag{11}$$

It is shown in [55] that $\frac{V[-x(1-q)]}{V[xq]}\geq\frac{1}{V[x]}$. Hence, (11) is satisfied if $\frac{1}{V[x]}\geq\frac{q}{1-q}$ and the sufficient condition for sub-regularity of $\varphi(x,q)$ on $[0,1]$ is to choose x such that $\frac{1}{V[x]}\geq\frac{q}{1-q}$.

Lemma 1. There exists $x^{*}\in R$ such that for $x\leq x^{*}$ the inequality (11) holds.

Proof.

We note that $V[x]$ is continuous, strictly increasing and positive. Moreover, $\frac{q}{1-q}$ has a positive value for any $0<q<1$. Hence, there exists $x^{*}\in R$ such that $V[x^{*}]=\frac{1-q}{q}$, and for $x\leq x^{*}$ the inequalities $\frac{1}{V[x]}\geq\frac{q}{1-q}$ and (9) hold. Hence, for $x\leq x^{*}$, as in Lemma 1, $\varphi(x,q)$ is sub-regular. It can therefore be concluded that $\varphi(x,q)\leq\Gamma(q)\leq 1$.

Moreover, from the definition of $\varphi(x,q)$, it is clear that for a given $\varepsilon>0$, there exists a positive constant a^{*} such that for $0<a\leq a^{*}$, we have $1-\varepsilon\leq\varphi(x,q)\leq\Gamma(q)\leq 1$. Hence, for a sufficiently small $\varepsilon>0$, Algorithm 1 converges to the minimum PIDS with probability 1 as $t\to\infty$. This completes the proof of Theorem 1. □

Theorem 2. Algorithm 1 converges to the minimal PIDS in the network after $I=\frac{log_{1-a}^{m\sqrt{1-\varepsilon}-1/\frac{1}{\Gamma}(1-a)^{N-1}-1}}{\frac{1-\varepsilon+q}{2}}$ iterations with probability not less than $1-\varepsilon$, where m and N are the average size and the number of PIDSs in the whole network respectively.

Proof.

To investigate an upper bound for the complexity, we focus on the worst case, in which all PIDSs greater than the minimal one, P_{min}, are selected first. The worst case can be divided into the two distinct phases of before and after the first proper selection; these two stages are referred to as the effort phase and the success phase, respectively.

In the effort phase, all of the selected PIDSs greater than P_{min} are selected. Without loss of generality, we suppose that these PIDSs are selected in decreasing manner; hence, each obtained PIDS is rewarded because it is greater than the previous one. Letting *i* be the iteration in which the maximal PIDS is obtained for the first time, all of the selected actions are rewarded during the first *i* iterations. We then have

$$p_i(N) \geq p_i(N-1)(1-a)^m, \tag{12}$$

where *N* is the total number of PIDSs in the network and *m* is their average size. The success phase is started in iteration *i,* in which $PIDS_{min}$ is selected for the first time. In the success phase, the probability of penalizing $PIDS_{min}$ is zero; thus, the change in the probability of selecting $PIDS_{min}$ is non-negative. More precisely, the probability of selecting $PIDS_{min}$ in this period increases if the corresponding actions are rewarded, and remains unchanged otherwise. Therefore, at each iteration of the success phase, the inequality

$$p_{PIDS_{min}}{}^j \geq \prod_{v_k \in PIDS_{min}} p_k{}^{j-1} + a\left(1 - p_k{}^{j-1}\right), \tag{13}$$

holds, where $p_{PIDS_{min}}{}^j$ is the probability of selecting $PIDS_{min}$ after *j* iterations in the success phase. Using (5) iteratively we obtain

$$p_{PIDS_{min}}{}^j \geq \prod_{v_k \in PIDS_{min}} (1-a)^j p_k^0 + \left[(1-a)^{j-1} + \cdots + (1-a) + 1\right]a, \tag{14}$$

where p_k^0 is the probability of selecting the action corresponding to v_k at the start of the success phase and is greater than $p_k(1-a)^{m-1}$ according to (12). Moreover, using the geometric series formula, (14) can be simplified to

$$p_{PIDS_{min}}{}^j \geq \prod_{v_k \in PIDS_{min}} (1-a)^j p_k (1-a)^{N-1} + \left[\frac{1-(1-a)^j}{a}\right]a \tag{15}$$

$$= \prod_{v_k \in PIDS_{min}} (1-a)^j [p_k(1-a)^{N-1} - 1] + 1$$

Because the algorithm starts from uniform probabilities, each p_k is initially equal to $\frac{1}{\Gamma}$ and we have

$$p_{PIDS_{min}}{}^j \geq \left((1-a)^j \left[\frac{1}{\Gamma}(1-a)^{N-1} - 1\right] + 1\right)^m.$$

Therefore, to ensure that the algorithm converges to $PIDS_{min}$ with probability greater than 1-ε, it is sufficient to let $\left((1-a)^j \left[\frac{1}{\Gamma}(1-a)^{N-1} - 1\right] + 1\right)^m \geq 1 - \varepsilon$.

Now, the minimum number of iterations in the success phase is equal to$i = log_{1-a}^{\sqrt[m]{1-\varepsilon}-1/\frac{1}{\Gamma}(1-a)^{N-1}-1}$.

We note that when an incorrect PIDS is chosen during the success phase, the probability of selecting $PIDS_{min}$ remains invariant; hence, *i* does not include the iterations in which the other PIDSs are selected. In the remainder of this section, we discuss how to compute the total number of iterations.

Let P_{ave} be the average probability of selecting $PIDS_{min}$ in the first k iterations. Because this probability is equal to *q* in the first iteration and reaches 1-ϵ after *i* iterations in the success phase, its average value is $P_{ave} = \frac{1-\varepsilon+q}{2}$. Conversely, assuming that *I* is the total number of iterations, it is clear that $i=IP_{ave}$ and hence $I = \frac{log_{1-a}^{\sqrt[m]{1-\varepsilon}-1/\frac{1}{\Gamma}(1-a)^{N-1}-1}}{\frac{1-\varepsilon+q}{2}}$. □

This completes the proof.

Remark 1. In Theorem 2, the maximum number of iterations necessary for Algorithm 1 to reach a proper solution is presented. Here, we study the computational complexity of each iteration to provide a proper bound for the computational complexity of the algorithm. The average degree among the nodes is O(2n/n)≃O(1). Hence, the total complexity in an iteration of the algorithm has the same order as one node. Moreover, the most time-consuming part of the algorithm is the selection of an action based on the action probability vector, which in the worst case takes $O(n^2)$. Consequently, the total complexity of the algorithm is not greater than $T_{max} = O(n^2\psi)$, where $\psi = \frac{log_{1-a}^{\sqrt[m]{1-\varepsilon}-1/\frac{1}{\Gamma}(1-a)^{N-1}-1}}{\frac{1-\varepsilon+q}{2}}$.

5 Experiments

In this section, we evaluate the performance of our proposed algorithm for several well-known real and synthetic networks, including Zachary's Karate Club (*Karate*) [56], American college Football (*Football*) [57], the Dolphins network (*Dolphins*) [58], Jazz musicians network (*Jazz*) [59] and BA-50 to 500 as synthetic networks. All of these networks are commonly used benchmarks in simulations. Table 1 shows the real and synthetic networks that were used in the experiments and their characteristics. As an example of a well-known synthetic network, we used the *Barabási-Albert* model (BA model), which is a scale-free network with heavy-tailed degree distribution [3]. In addition, we assume that all undirected datasets are double weighted and directed for influence maximization, in which the weight of the edge (*u*, *v*) is 1/deg(*v*), where *deg*(*v*) is the in-degree of node *v*.

Table 1. Real and synthetic networks used in simulations.

Network	Nodes	Edge	Description
Karate	34	78	Zachary's Karate Club Network
Dolphins	62	159	Networks of Dolphins

Football	115	613	Network of American College Football Teams
Jazz	195	5484	Network of American musicians
BA-[50-500]	[50-500]	[200-2000]	Synthetic Scale-Free Networks

Sections 5.1 and 5.2 present experimental results corresponding to the MPIDS and influence maximization problems respectively. All experiments with the algorithm were launched on a system with a hardware configuration of Intel® Core i5 2.67 GHz and 4 GB RAM. The linear learning algorithm L_{R-I} was used for the learning automaton, and the learning parameter 0.001 was applied in all of the experiments.

5.1 Experimental Result for the MPIDS Problem

To demonstrate the performance of the proposed positive influence dominating set algorithm, several experiments were conducted on well-known real and synthetic networks. The termination condition of the proposed algorithm is when either the number of iterations reaches 10,000 or the value of entropy is less than 0.1. The entropy value is computed using Equation (3), considering the learning automata of the nodes that are members of candidate PIDS at specific iterations.

5.1.1 Different Learning Rates

In any learning automaton-based algorithm, the selection of the proper learning rate a is a difficult task. In this experiment, we study the effect of choosing different learning rates on the performance of the proposed algorithm. In Table 2, the sizes of the obtained MPIDS are represented for different learning rates and datasets. Each experiment is conducted 30 times, and the results are reported based on the average and standard deviation values. From Table 2, we can conclude that the performance of the algorithm is significantly sensitive to the value of the learning rate a; in in most cases, the lower the learning rate a, the lower the size of the obtained MPIDS. Therefore, in the rest of the experiments, the value of the learning rate a is assumed 0.001. The best result for each dataset is highlighted in boldface.

Table 2. Average results [±standard deviation] for Algorithm 1 in terms of obtained minimum PIDS.

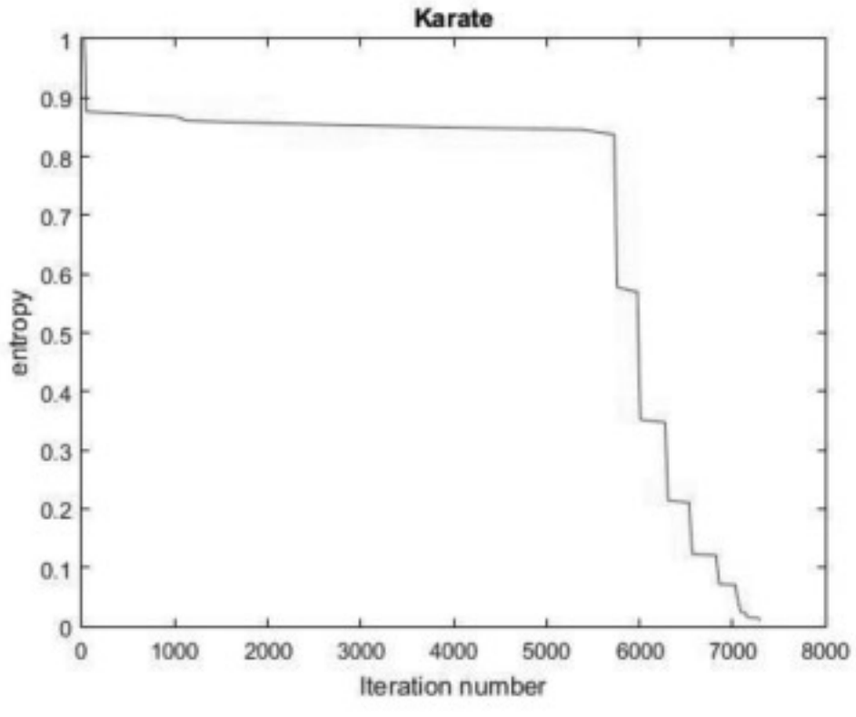
Learning Rate(a)	Karate	Dolphins
0.001	**19.00±1.84**	47.08±2.05
0.002	19.00±1.91	**45.00±2.07**
0.003	19.00±1.91	45.00±2.11
0.004	19.00±2.11	46.00±3.02
0.005	19.00±2.21	46.00±3.09
0.006	19.00±2.22	46.00±3.11
0.007	19.00±2.23	45.00±3.89
0.008	19.00±2.25	46.00±3.12
0.009	19.00±2.33	45.00±4.12
0.01	19.00±2.35	45.00±4.99
0.02	19.01±3.51	47.00±3.02
0.03	19. 92±3.55	46.00±3.05
0.04	19. 73±3.56	46.01±3.41
0.05	19. 29±3.11	46.03±3.59
0.06	19. 17±3.05	45.27±5.11
0.07	19.23±3.19	47.23±3.33
0.08	19.23±3.22	46.25±3.65
0.09	19.46±4.56	46.50±3.99
0.1	19.93±4.66	47.34±4.12

BA-500	BA-400	BA-300	BA-200	BA-100	Football	Jazz
444.00±5.22	**317.00±0.05**	242.03±1.21	**166.00±2.51**	**72.00±1.22**	89.00±1.55	**176.00±2.22**
444.38±5.45	317.00±0.12	**242.00±1.02**	166.00±3.99	72.50±2.14	**88.00±1.99**	183.00±3.31
446.00±6.25	318.00±1.02	244.02±1.25	166.02±4.25	72.00±2.19	90.00±2.01	180.00±4.02
445.09±5.25	317.00±0.34	242.00±2.51	167.04±4.41	72.05±2.33	90.50±2.09	180.00±4.11
451.00±5.21	317.00±0.44	244.02±3.14	170.00±3.21	73.00±2.02	91.00±1.99	181.00±4.91
446.00±3.56	319.00±0.65	244.00±3.24	168.00±3.02	72.00±2.51	92.00±1.88	182.00±4.99
446.00±4.25	319.00±0.75	245.00±2.99	169.00±4.11	72.00±2.55	90.00±2.22	180.00±4.55
448.00±5.25	318.00±1.85	245.00±2.74	167.00±4.12	77.00±4.21	91.00±2.01	181.00±5.01
451.00±5.31	319.00±2.22	249.00±4.25	169.00±4.22	74.00±3.25	90.00±2.34	180.00±5.05
451.00±5.51	322.00±2.51	246.00±4.33	166.00±5.12	73.00±2.18	88.28±5.22	179.00±4.25
460.00±7.21	319.03±2.65	249.05±4.45	168.00±4.21	74.00±3.22	93.00±3.58	178.00±4.33
455.00±6.54	326.08±4.16	251.00±4.32	171.00±3.21	77.00±5.14	93.07±3.84	180.00±5.08
461.02±6.66	329.59±4.52	252.02±5.02	168.04±4.51	77.01±5.21	91.01±4.02	179.00±4.62
459.70±7.21	332.13±5.04	249.64±5.21	170.14±4.22	77.08±7.35	99.03±5.77	179.00±4.77
462.83±6.58	333.52±5.24	251.38±6.28	176.34±5.41	77.18±7.36	92.18±3.54	178.00±3.77
461.61±5.95	338.92±5.59 1	252.75±6.55	177.41±5.51	77.36±7.38	91.38±5.55	179.00±6.19
460.78±7.25	334.05±5.95	252.75±6.58	178.34±5.81	77.73±7.55	97.28±6.45	179.00±6.25
463.19±7.11	334.12±5.99	252.53±6.67	179.66±5.91	77.72±5.65	92.72±4.22	178.00±4.025
467.19±7.24	336.25±6.38	255.00±7.65	171.62±5.67	78.56±6.45	93.44±4.25	181.00±5.95

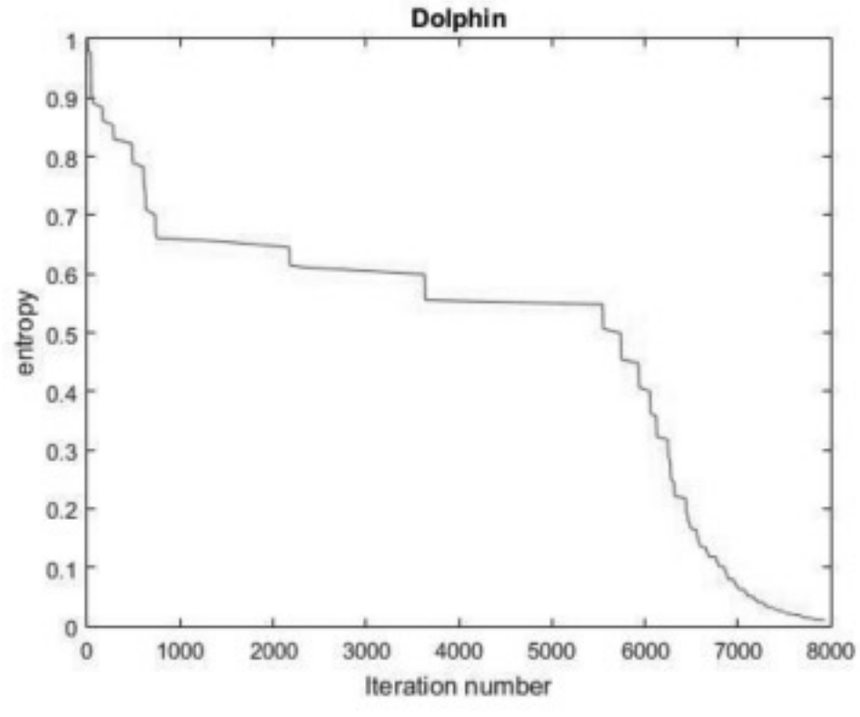
5.1.2 Convergence Behavior of the Proposed Algorithm

In this experiment, we investigate the convergence behavior of the proposed algorithm. For this purpose, the entropy value versus the iteration number is plotted in Figure 4. The uncertainty for learning automata in selecting the next group of actions is indicated by the entropy; the greater the entropy value, the higher the degree of uncertainty of the next action set, which means that the learning automata have no useful data for obtaining a proper result and the action selected is equivalent to a random selection. Conversely, a lower uncertainty value confirms that the learning automaton selects a proper action set with a high probability.

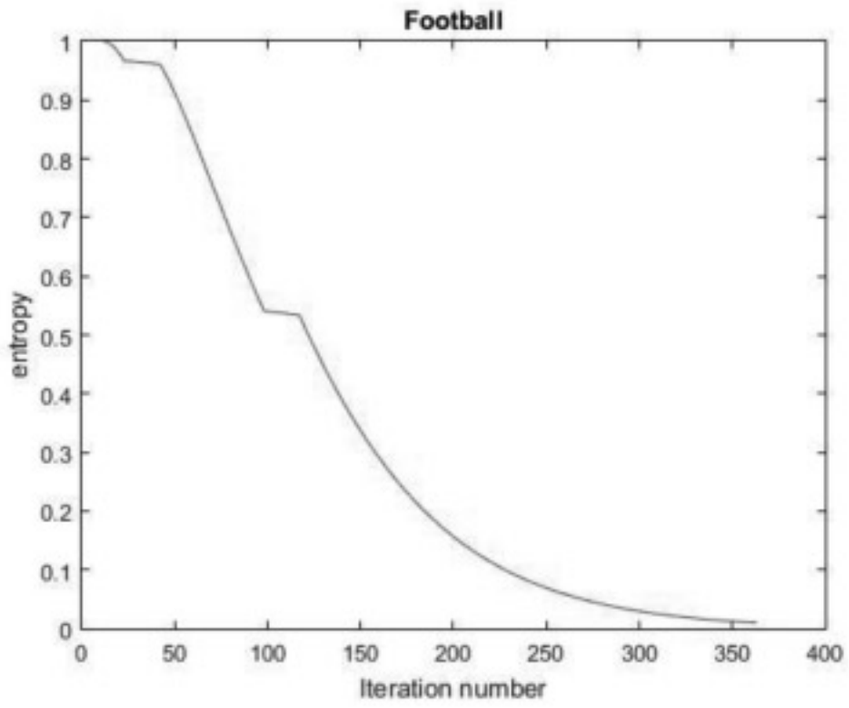
As shown in Figure 4, the entropy value gradually decreases with the increase in iteration number. We note that the smooth reduction in entropy value shows that the algorithm converges to an optimal action. In addition, the trend in the reduction in entropy value is considerably different in different networks. This trend is confirmed by the entropy plots of Figure 4. More accurately, the reduction trend differs from one network to another, meaning that the entropy value depends upon the topology of the networks.



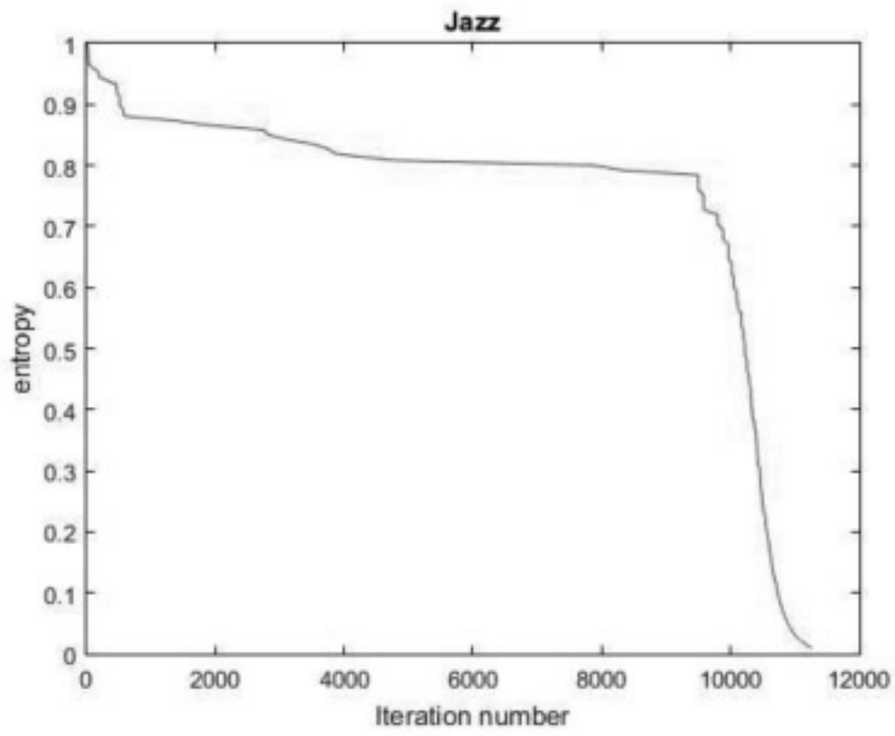
a) Karate



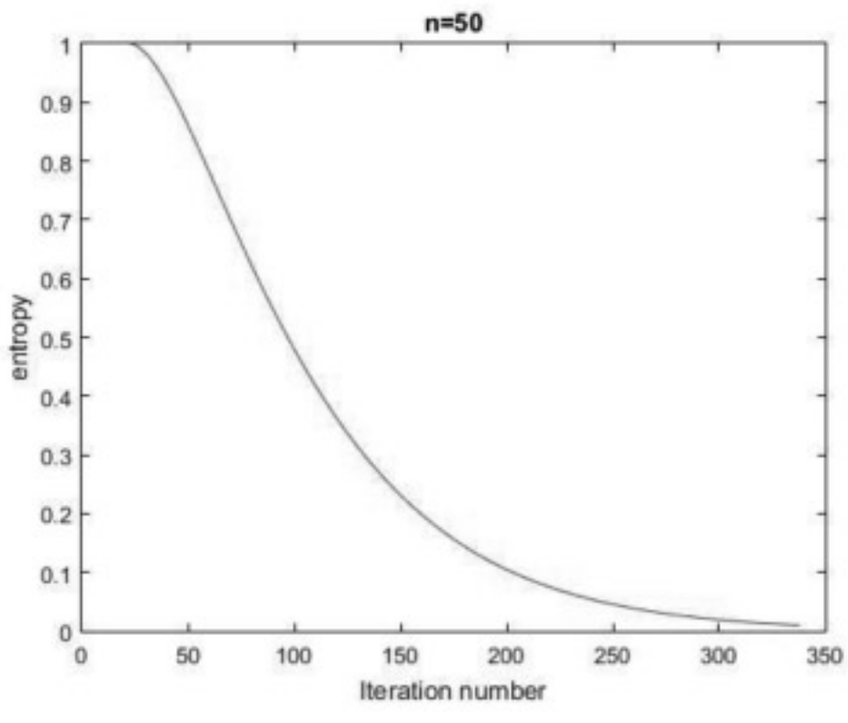
b) Dolphins



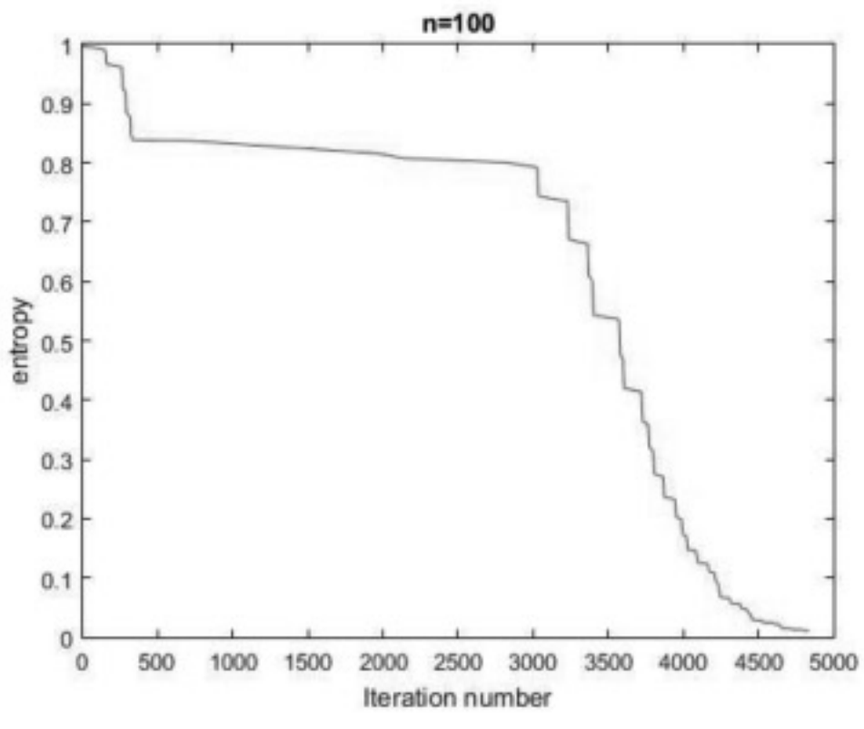
c) Football



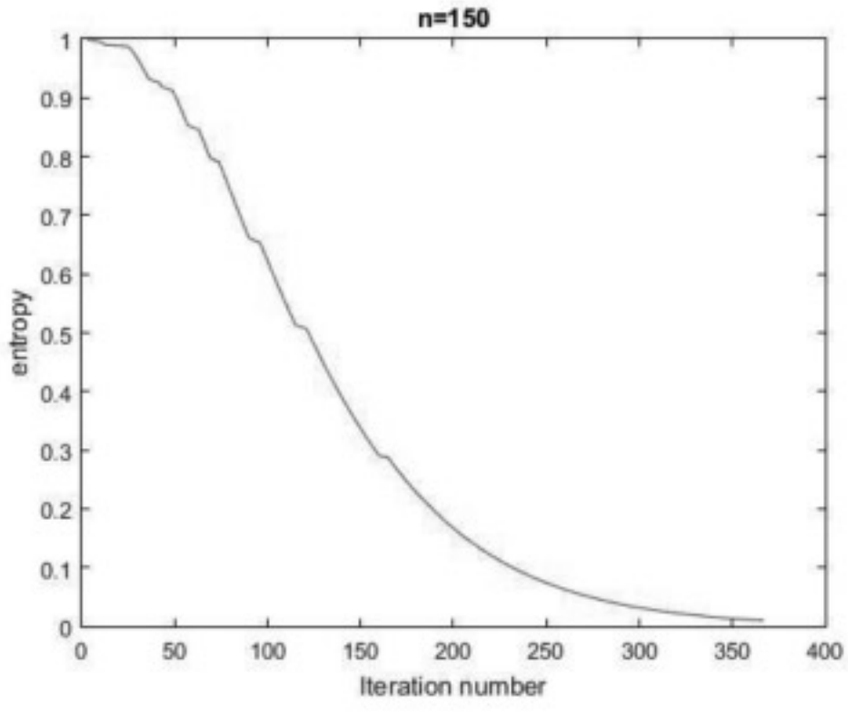
d) Jazz



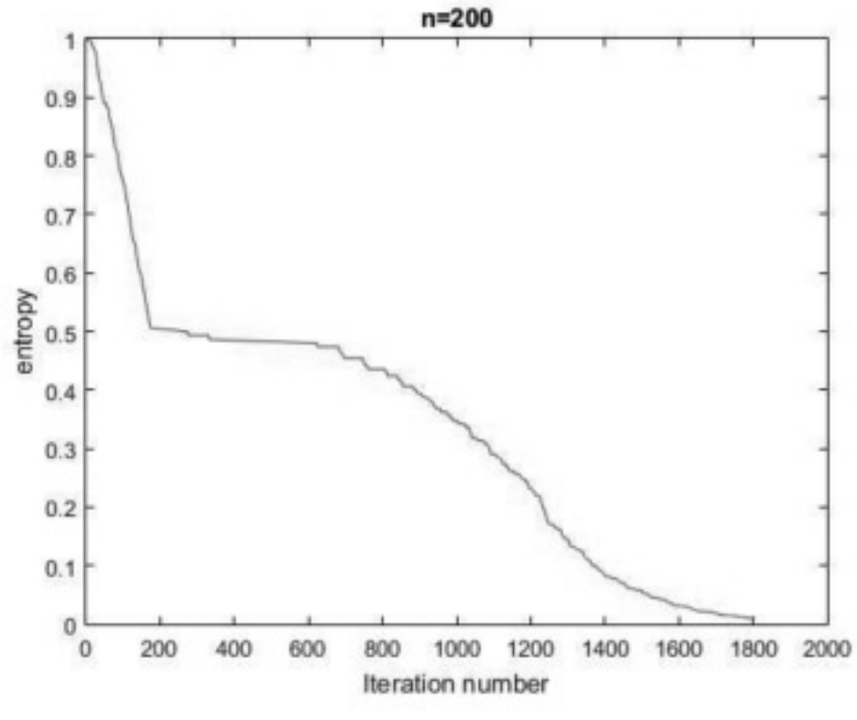
e) BA-50



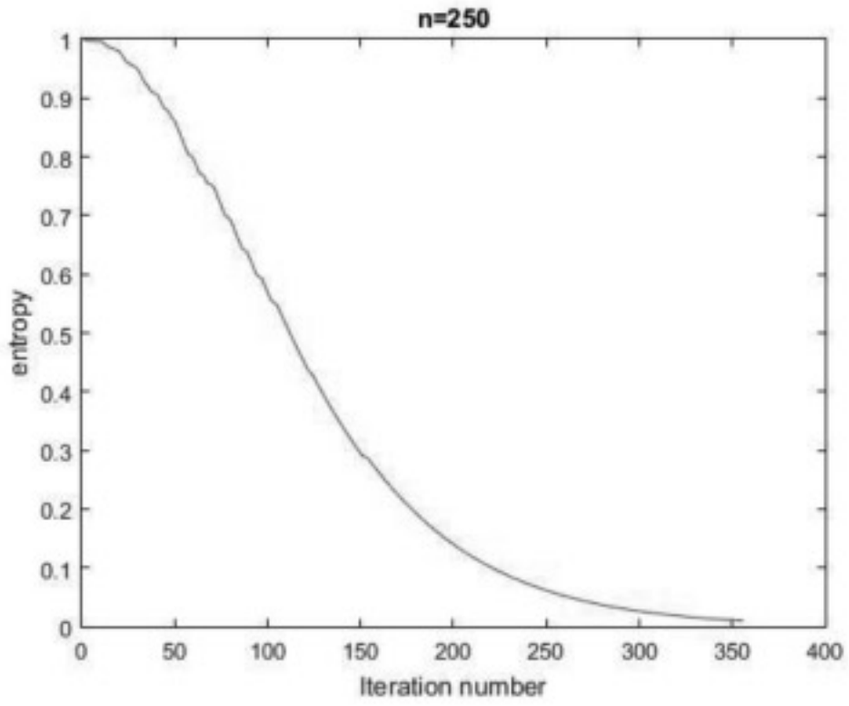
f) BA-100



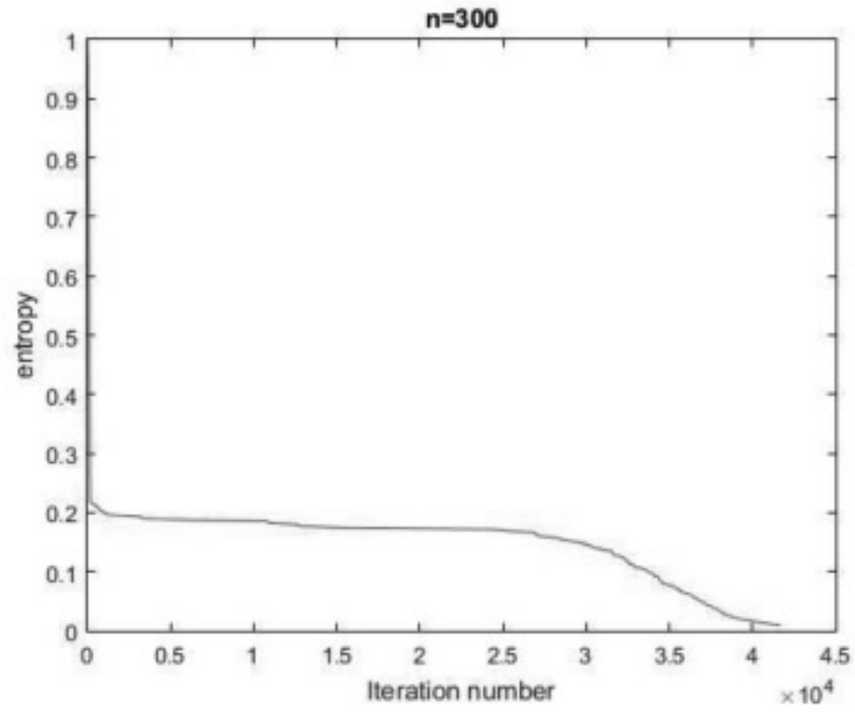
g) BA-150



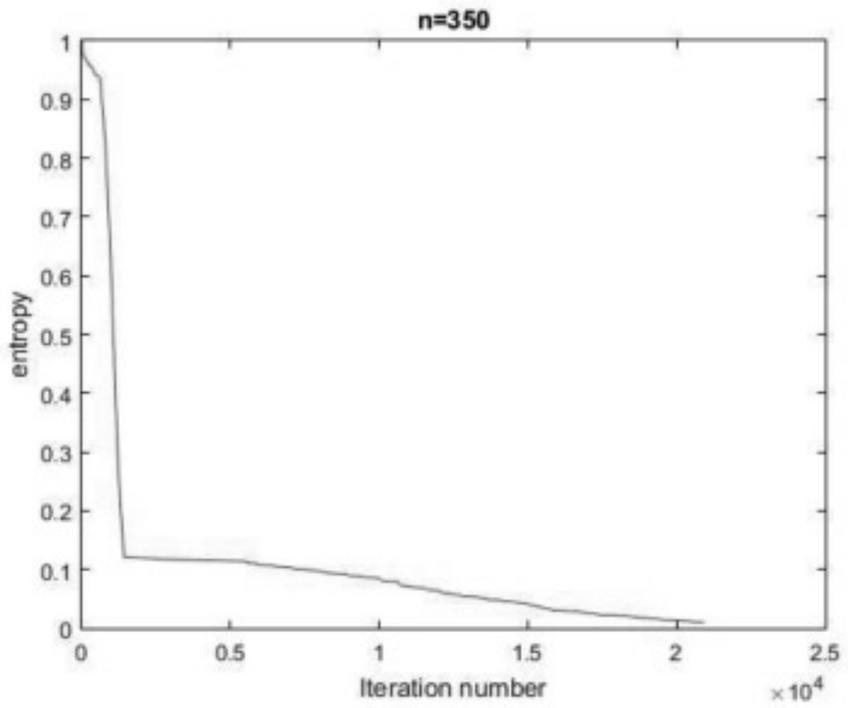
h) BA-200



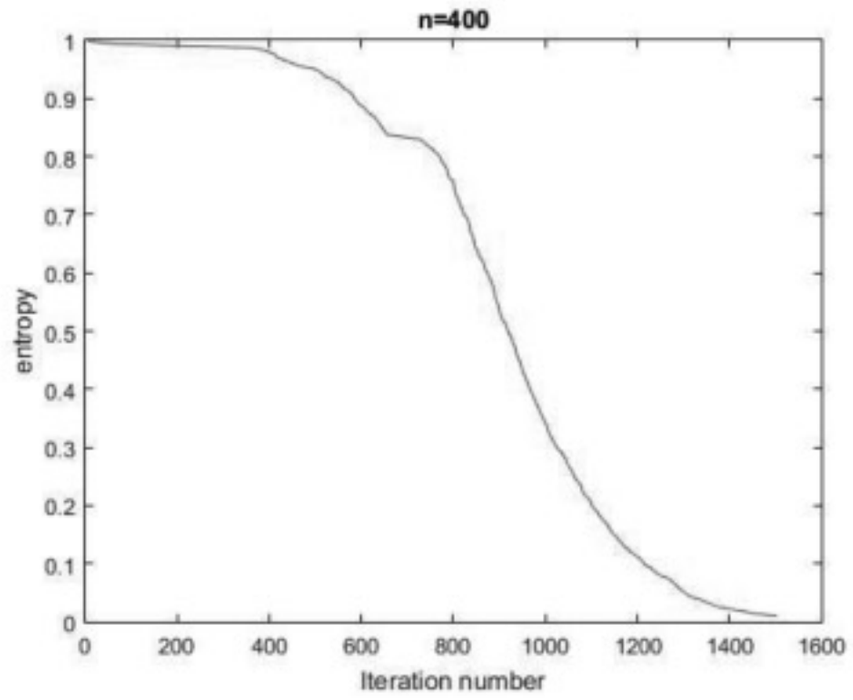
i) BA-250



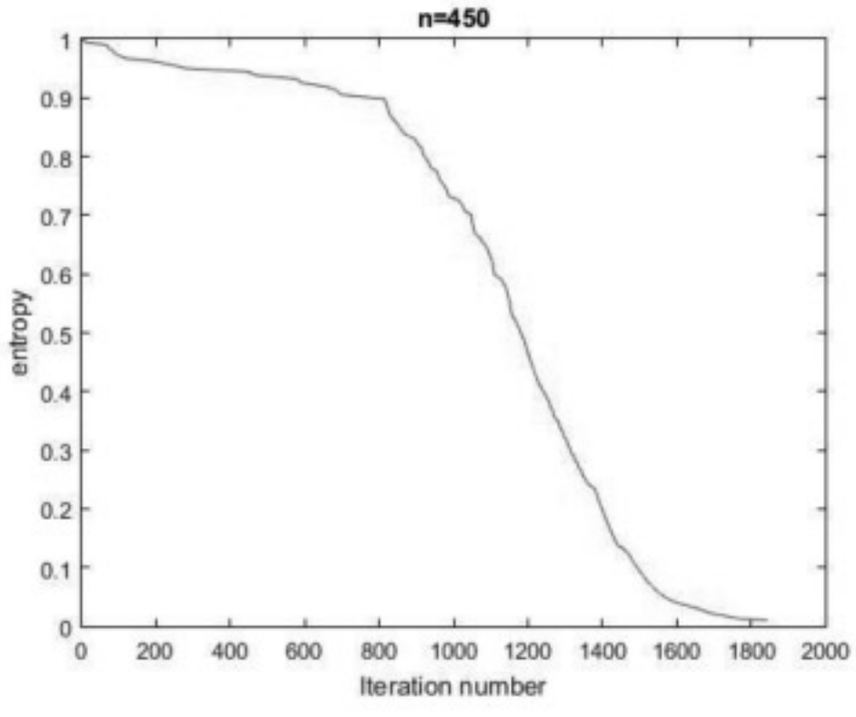
j) BA-300



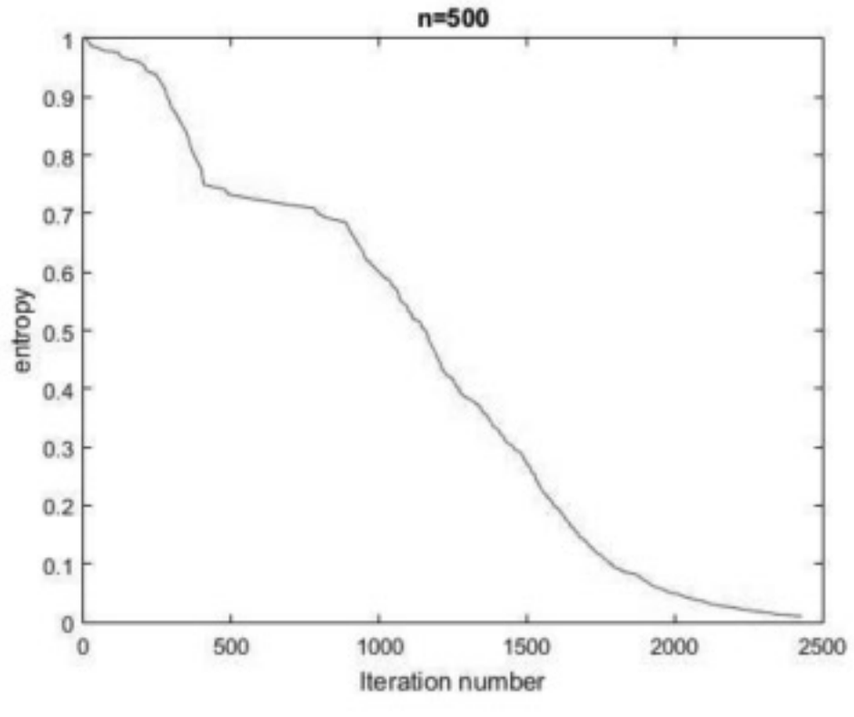
k) BA-350



l) BA-400



m) BA-450



n) BA-500

Figure 4. Entropy value of the proposed algorithm on the experimental datasets.

5.2 Experimental Results for Influence Maximization

In our proposed algorithm, we initially discover the MPIDS for a given network and then use the obtained results for influence maximization. Several algorithms and heuristics have been investigated for influence maximization and are compared below.

5.2.1 Total Influence Spread by Heuristics

To study the performance of the proposed algorithm, we conducted several experiments to compare the influence spread obtained by our proposed algorithm with High-degree [4], PageRank [18], HITS [19], CELF [21], CELF++ [22], K-Shell [60] and Pareto-Shell [61]. The computational points below were considered in applying these algorithms:

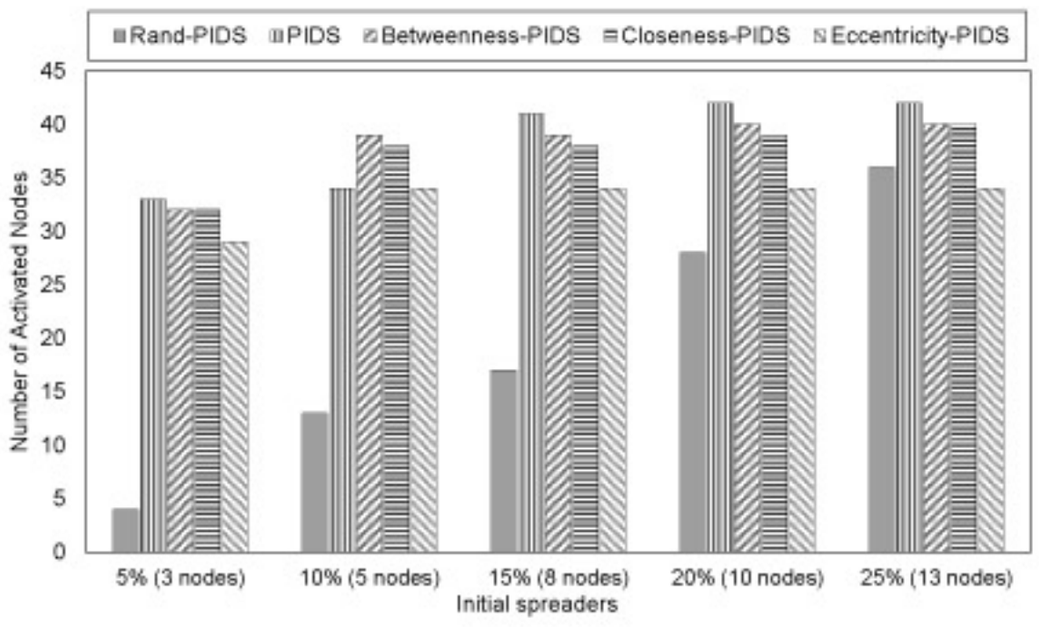
- For PageRank, we selected the *k* number of highest-ranked nodes as initial seed nodes and we used $\varepsilon=10^{-4}$ as a stopping condition for convergence of the PageRank algorithm.
- In HITS, we set the stopping criterion at 1.02 for selecting the *k* initial seed nodes.
- CELF was run 10,000 times to evaluate and estimate the influence spread of the seed set.

The descriptions of the algorithms used to provide comparisons are presented in Table 3.

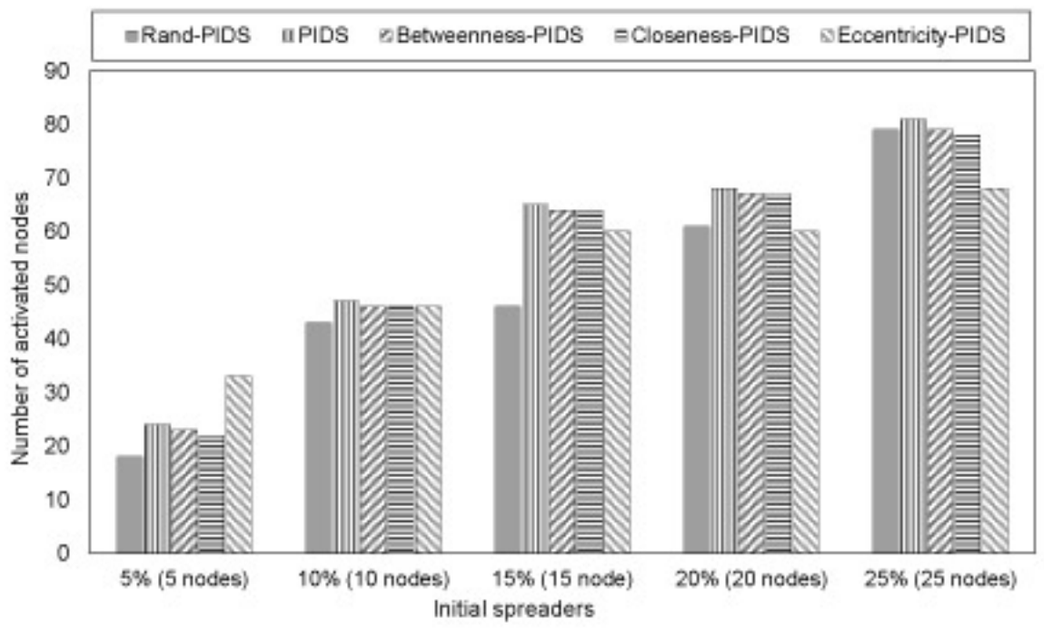
Table 3. Description of the Algorithms used for experimental comparison.

Algorithm	Description
High-degree [4]	A heuristic approach that selects *k* nodes with maximum degree.
HITS [19]	Hyperlink-Induced Topic Search (HITS; also known as hubs and authorities) is a link analysis algorithm that rates Web pages.
PageRank [18]	This algorithm is used for ranking webpages according to their importance in a web graph.
CELF [21]	A greedy algorithm that executes CELF using a Monte Carlo simulation.
CELF++ [22]	An improved version of CELF.
K-Shell [60]	This algorithm is an improved version of MDD [1] that considers both the residual degree and exhausted degree.
Pareto-Shell [61]	An improved version of K-Shell algorithm with the Pareto optimal set consisting of non-dominated spreads, with the ratio of high out-degree to in-degree and high in-degree.

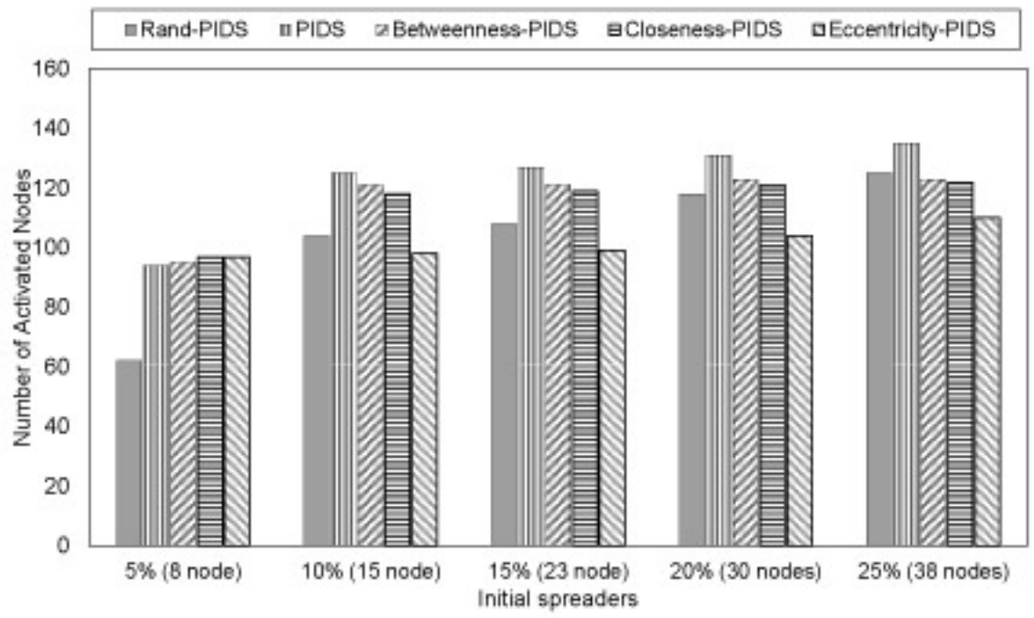
This experiment was conducted to study the effect of different selections of initial active nodes based on different criteria on the performance of the proposed algorithm. To this end, we studied the effect of the selection of various numbers of initial nodes on the active seed nodes obtained by the proposed algorithm. We initially obtained the MPIDS by Algorithm 1 and calculated the activated nodes based on the following policy. Different criteria are used, including Random node selection, High-degree nodes, High-betweenness nodes, High-closeness nodes and High-eccentricity nodes. The initial activated seed nodes are assumed 5%, 10%, 15%, 20% and 25% of the whole networks. In each case, a specific percentage of the nodes that are the best according to one of the assumed criteria are selected as the initial activated nodes. Figure 5 shows the numbers of activated nodes for different numbers of the initial active nodes for the different criteria, in which x-axis represents the number of initial activated nodes and y-axis represents the total numbers of activated seed nodes. In these figures, the short forms of the criteria are used as follows: Rand-PIDS, PIDS, Betweenness-PIDS, Closeness-PIDS and Eccentricity-PIDS stand for Random, High-degree, High-Betweenness, High-Closeness and High-Eccentricity criteria.
Compared with other algorithms, our proposed algorithm performs as well as other heuristics in terms of High-degree PIDS. We note that the term High-degree PIDS refers to the PIDS in which the nodes are sorted based on their degree. In some real networks, such as Dolphins and Football, the High-degree PIDS naturally has a high influence spread because these networks consist of many communities with High-degree nodes; hence, the greatest influence spread is expected to be obtained by selecting a High-degree node. In addition, in other datasets such as BA-50, BA-150, BA-250, Karate and Jazz, the number of activated nodes at the end of the proposed algorithm is as close as in the other algorithms.



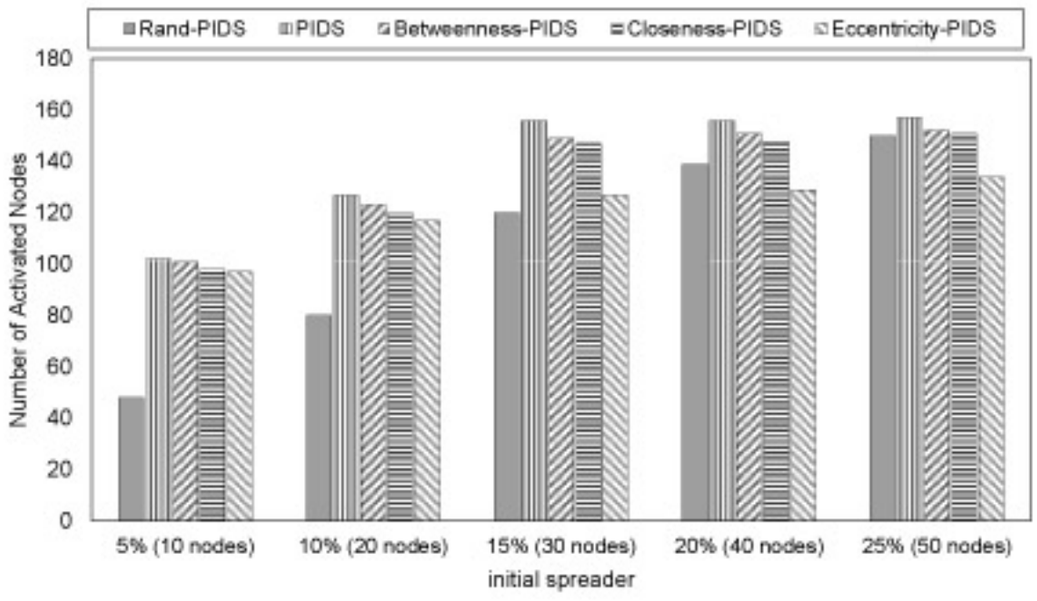
a) BA-50



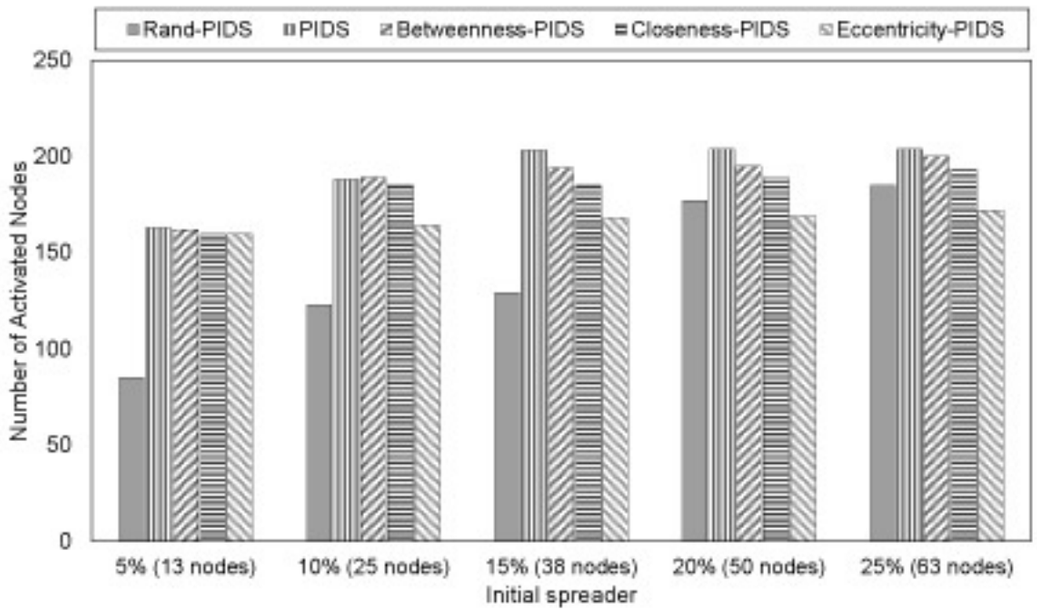
b) BA-100



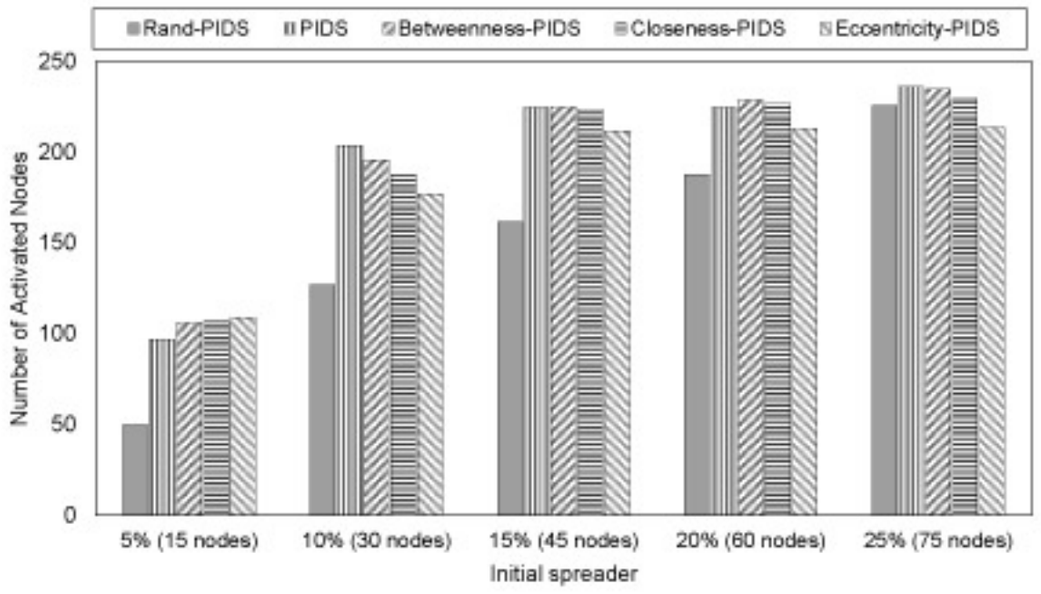
c) BA-150



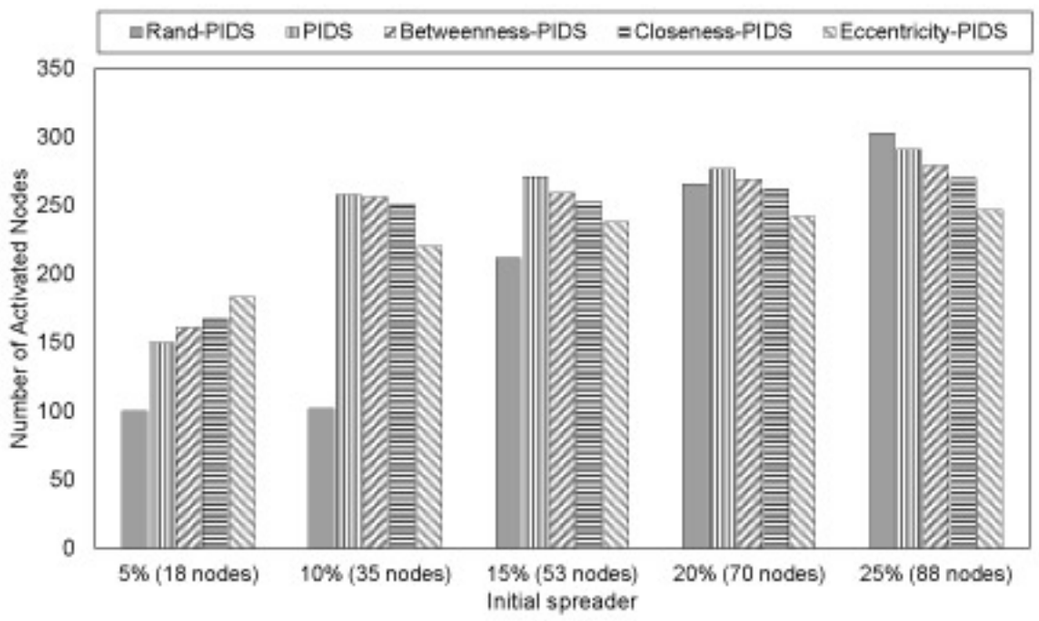
d) BA-200



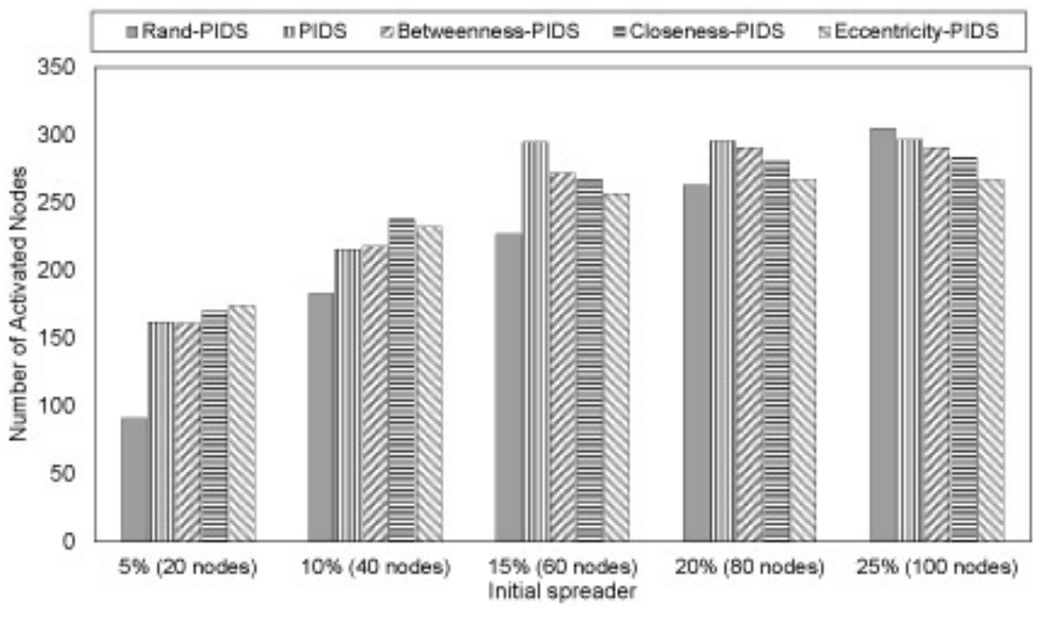
e) BA-250



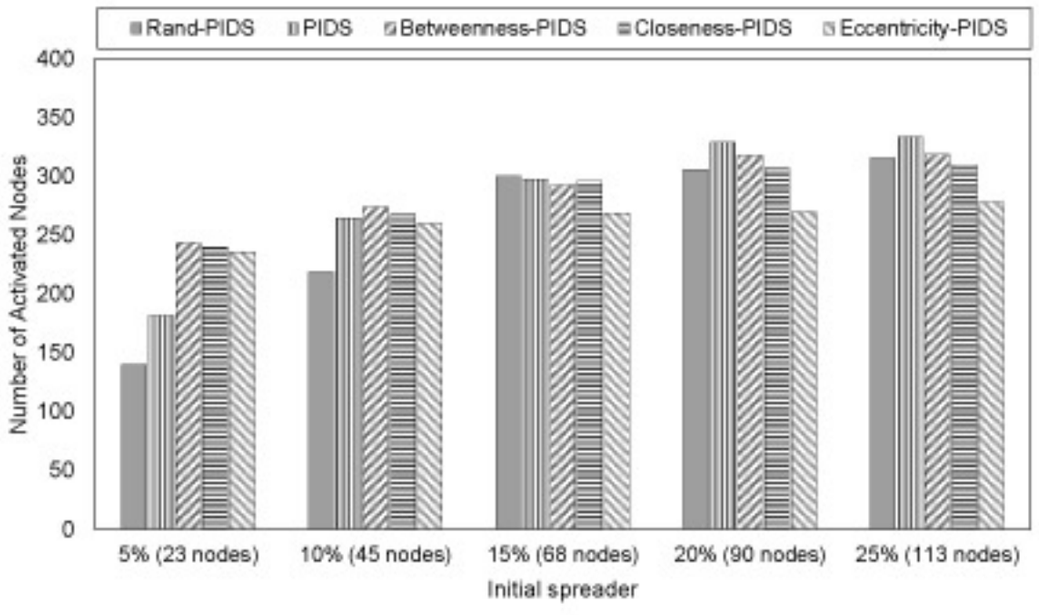
f) BA-300



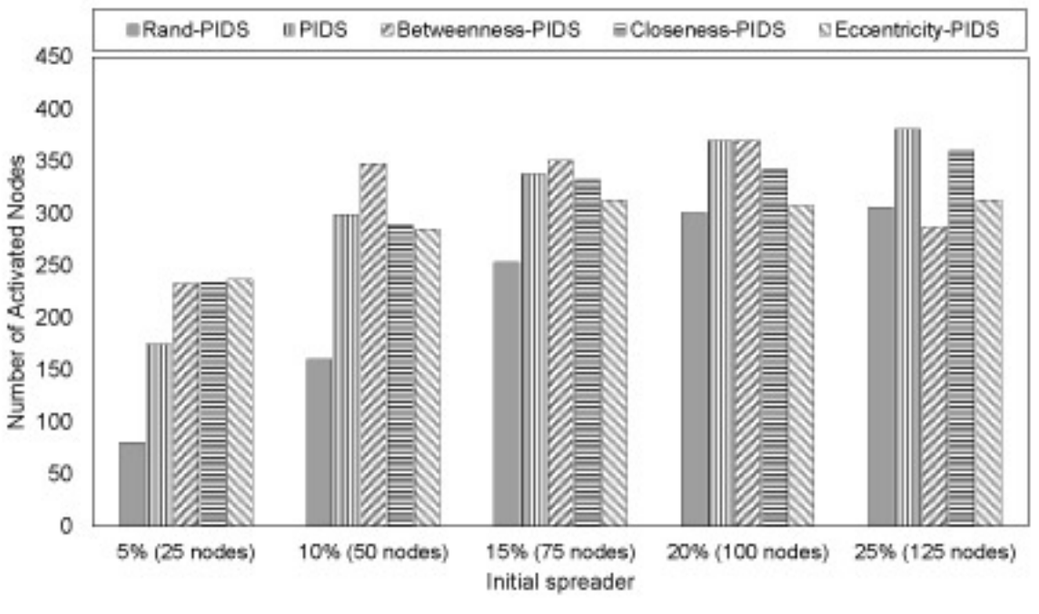
g) BA-350



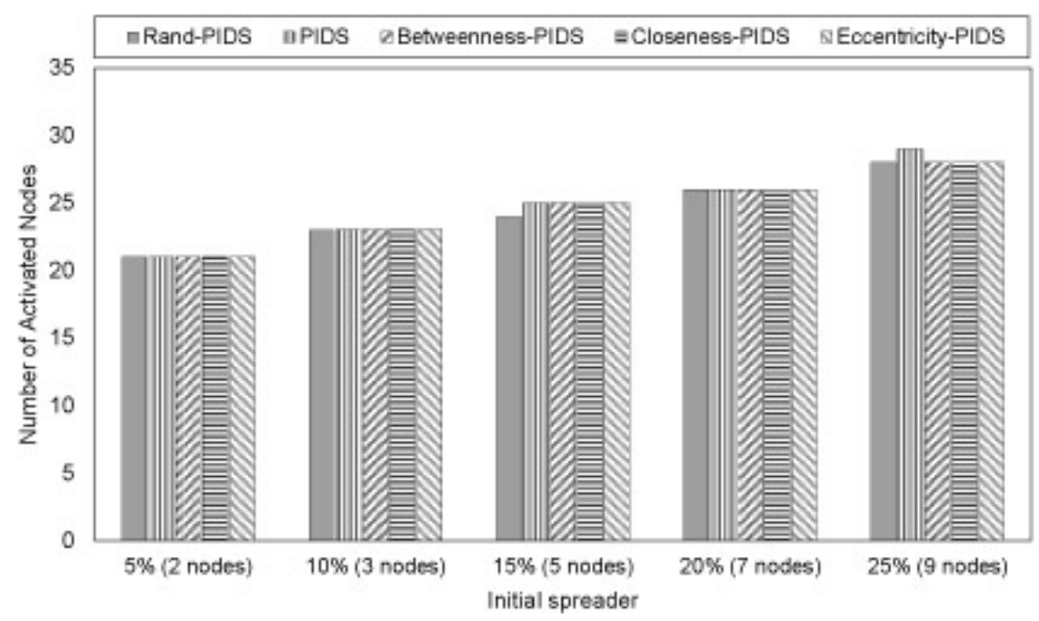
h) BA-400



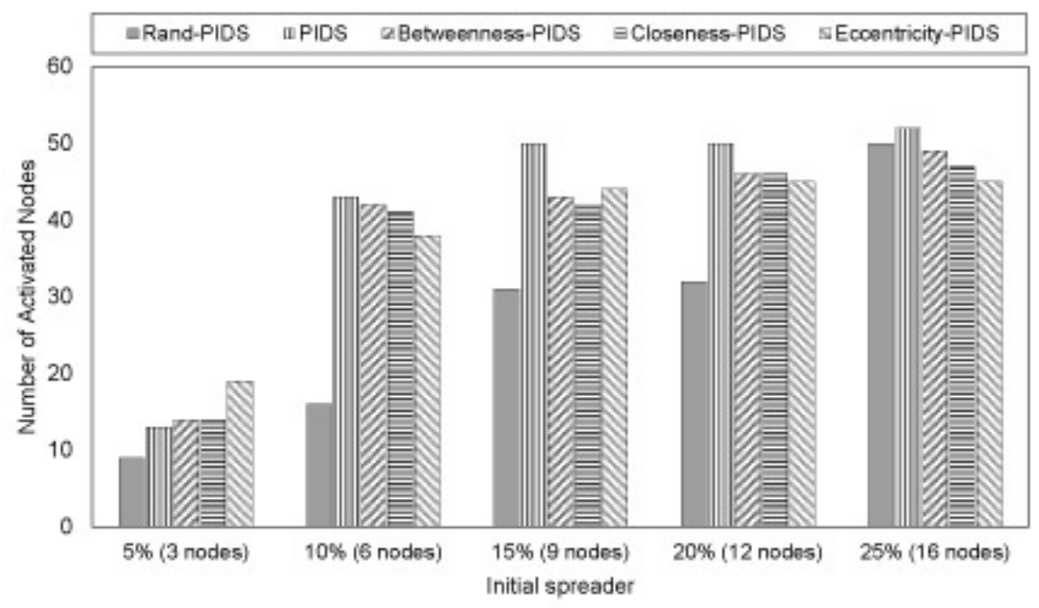
i) BA-450



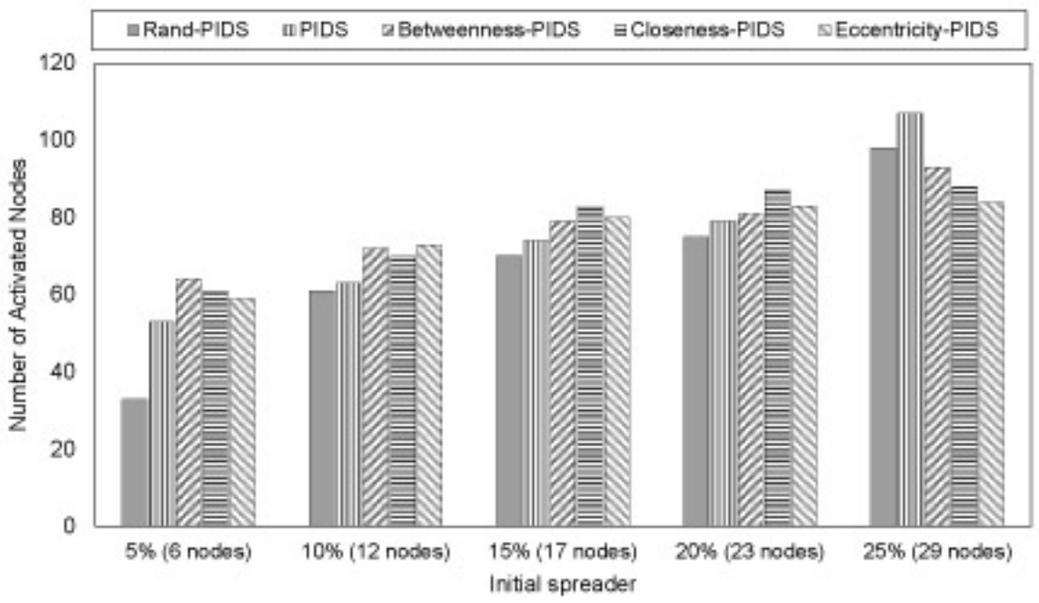
j) BA-500



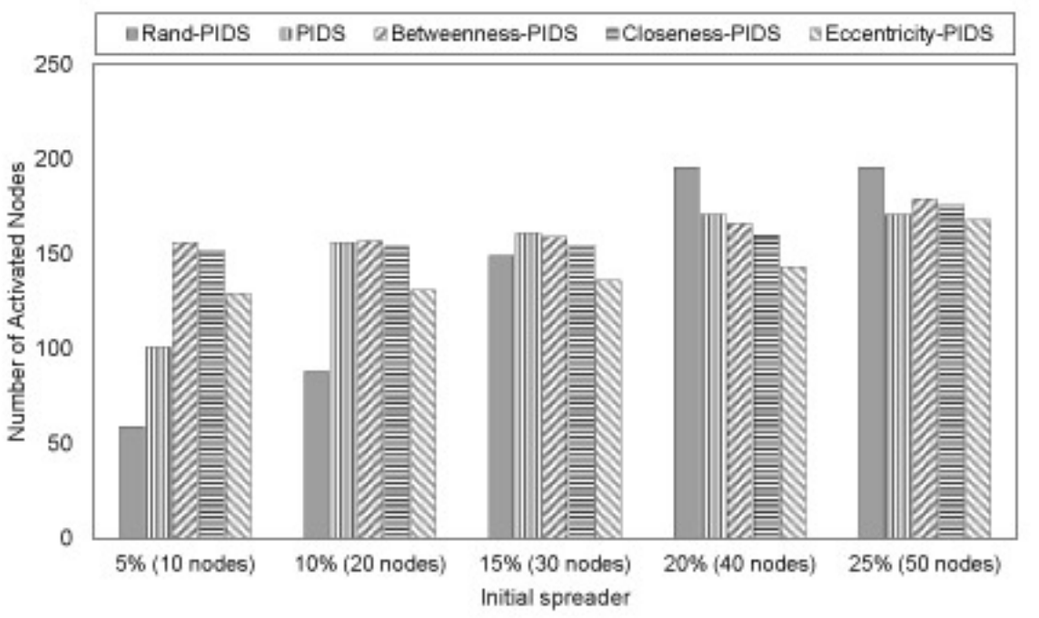
k) Karate



l) Dolphins



m) Football



n) Jazz

Figure 5. Number of activated nodes using different centrality and initial numbers of activated nodes.

To find the best-performing algorithm among all PIDS variants (Random, High-degree, High-Betweenness, High-Closeness and High-Eccentricity), a *Tukey-Kramer* multiple comparison [62] [60] has been applied over all test networks. Table 4 shows the result of multiple comparisons in the form of (*b*, –*w*). For each cell (*row*, *col*) of table 4, *b* and –*w* represent the number of test networks for which $PIDS_{row}$ is significantly better than and worse than $PIDS_{col}$, respectively. If the subtraction of *b* and *w* (boldface values) is a positive number, then $PIDS_{row}$ is better than the $PIDS_{col}$ and vice versa. When this value is equal to zero, none of the algorithms is superior to any of

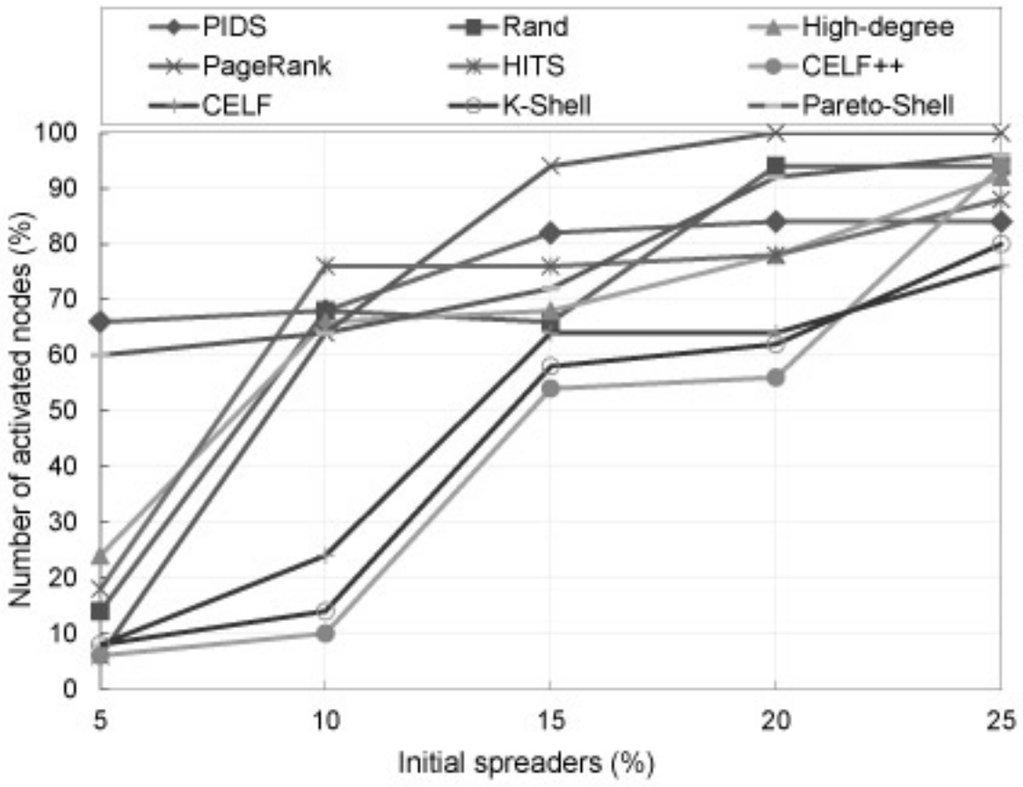
the others. The last column of Table 4 indicates the overall superiority of each algorithm over the other algorithms, which is obtained by summing the boldface values of each row. From the results of table 4, one can conclude that PIDS-Rand is the worst performing algorithm, being outperformed by all of the other PIDS variants in most cases and PIDS is the best performing algorithm on the set of benchmark functions that is not outperformed by any other PIDS variants in most cases. According to the obtained results and to the simplicity computation of degree, in the rest of this paper, we choose High-degree as a criterion for selecting MPIDS for studying the performance of PIDS in comparison with that of other algorithms.

Table 4. Multiple comparison of influence spread of algorithms based on the Turkey-Kramer method.

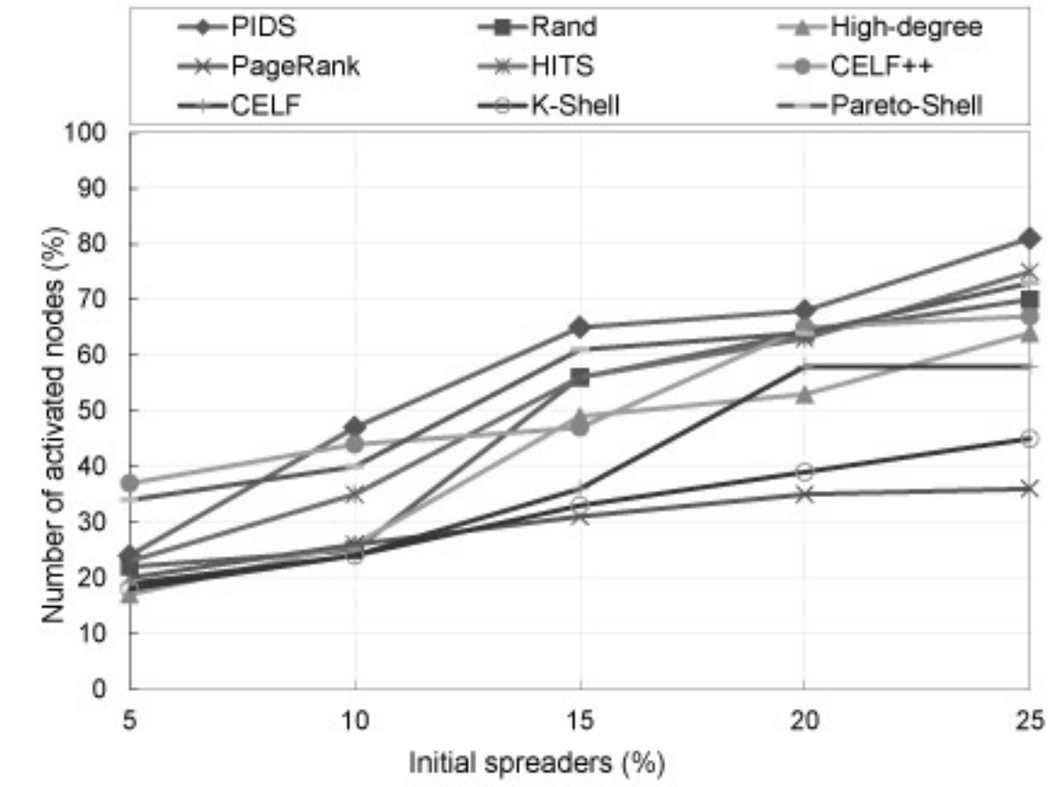
Algorithms	Rand-PIDS	PIDS	Betweenness-PIDS	Closeness-PIDS	Eccentricity-PIDS	Score
Rand-PIDS	0	(0, -42) **-42**	(+1, -38) **-37**	(+2, -38) **-36**	(+14, -29) **-15**	-130
PIDS	(+42, 0) **+42**	0	(+12, -6) **+6**	(+22, -8) **+14**	(+38, -7) **+31**	+93
Betweenness-PIDS	(+38, -1) **+37**	(+6, -12) **-6**	0	(+6, -5) **+1**	(+40, -6) **+34**	+66
Closeness-PIDS	(+38, -2) **+36**	(+8, -22) **-14**	(+5, -6) **-1**	0	(+33, -2) **+31**	+52
Eccentricity-PIDS	(+29, -14) **+15**	(+7, -38) **-31**	(+6, -40) **-34**	(+2, -33) **-31**	0	-81

5.2.2 Comparison with Other Algorithms

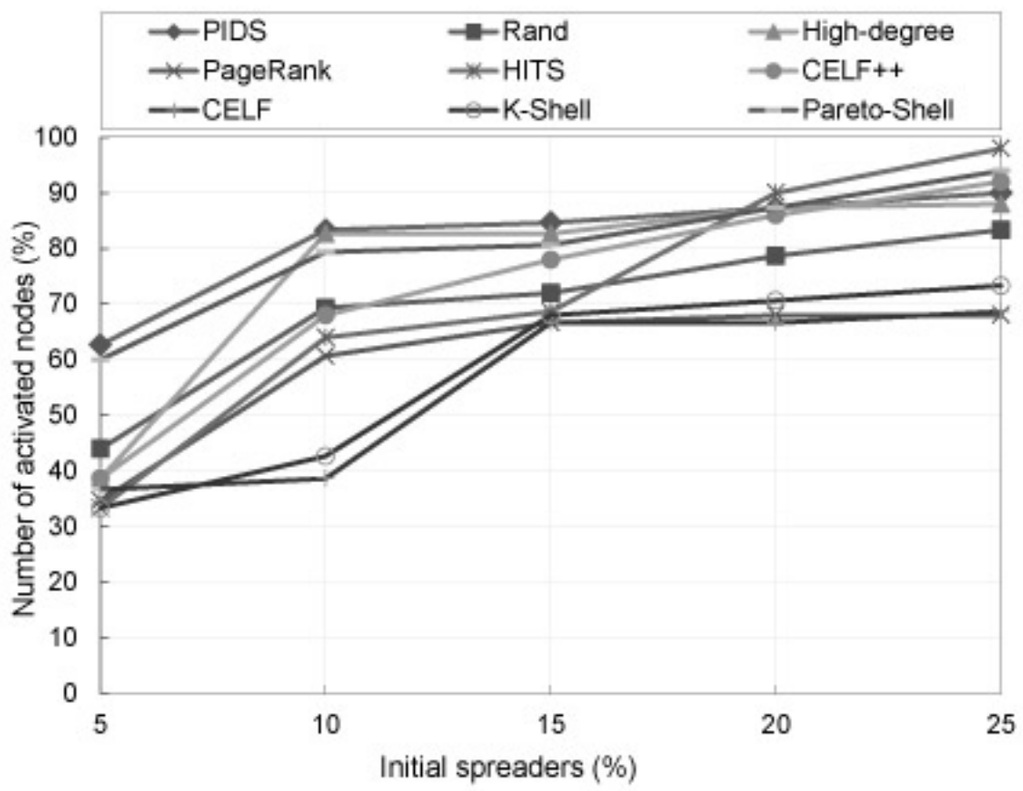
This experiment was conducted to compare the performance of the proposed algorithm (*PIDS*) with other algorithms (*Rand*, *High-degree* [4], *PageRank* [18], *HITS* [19], *CELF* [21], *CELF++* [22], *K-Shell* [60] and *Pareto-Shell* [61]) in terms of the number of activated nodes (influence spread) for different size of initial spreaders (seed nodes). Figure 6 shows the influence spread of algorithms on the test networks for different size for initial spreaders from 5 to 25% with a 5% interval. The results show that for all of the test networks, increasing the size of the initial spreaders results in increasing the total number of activated nodes at the end of the diffusion process. As shown in figure 6, *PIDS* activates more nodes than do other comparing algorithms for most synthetic networks. However, for some of real networks (*i.e.*, karate and jazz), *High-degree* and *PageRank* activate more nodes than does *PIDS,* meaning that PIDS provides important nodes (MPIDS) as initial spreaders (seed nodes) for the diffusion process, particularly in synthetic networks. This behavior might be due to the structural properties of these networks. In dense or modular (clustered) networks, *High-degree* and *PageRank* strategies cannot incorporate the fact that many of the most important nodes (*i.e.*, highest-degree or high PageRank) might be clustered or have several common neighboring nodes, so that targeting all of them is unnecessary. Conversely, some important (i.e., highest-degree or high PageRank) nodes can be ignored in MPIDS due to properties of MPIDS for existing common neighboring nodes. Therefore, it seems that PIDS is suitable for sparse networks and *High-degree* and *PageRank* strategies are suitable for dense or modular networks.



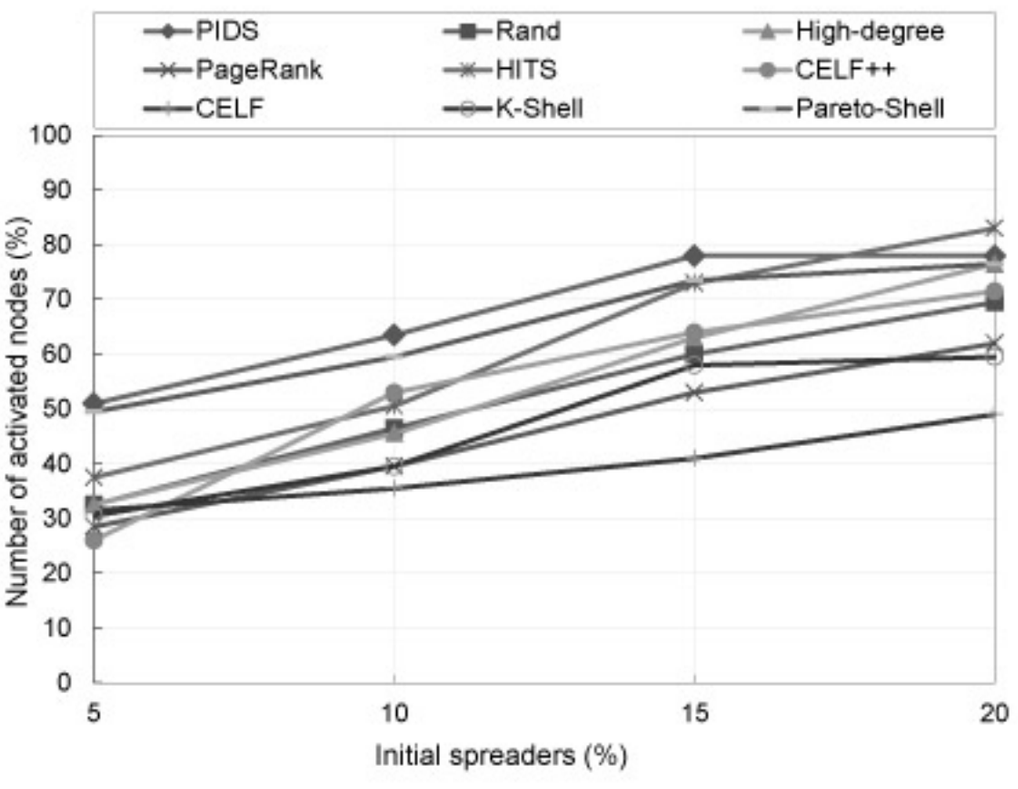
a) BA-50



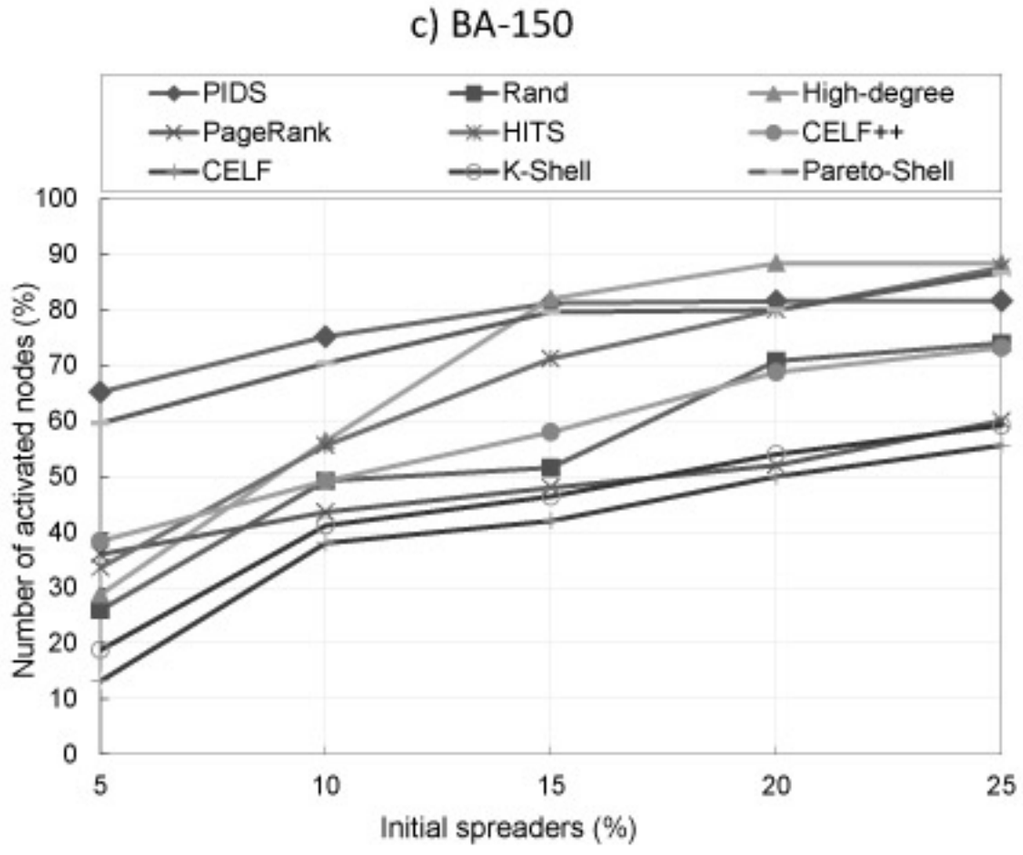
b) BA-100



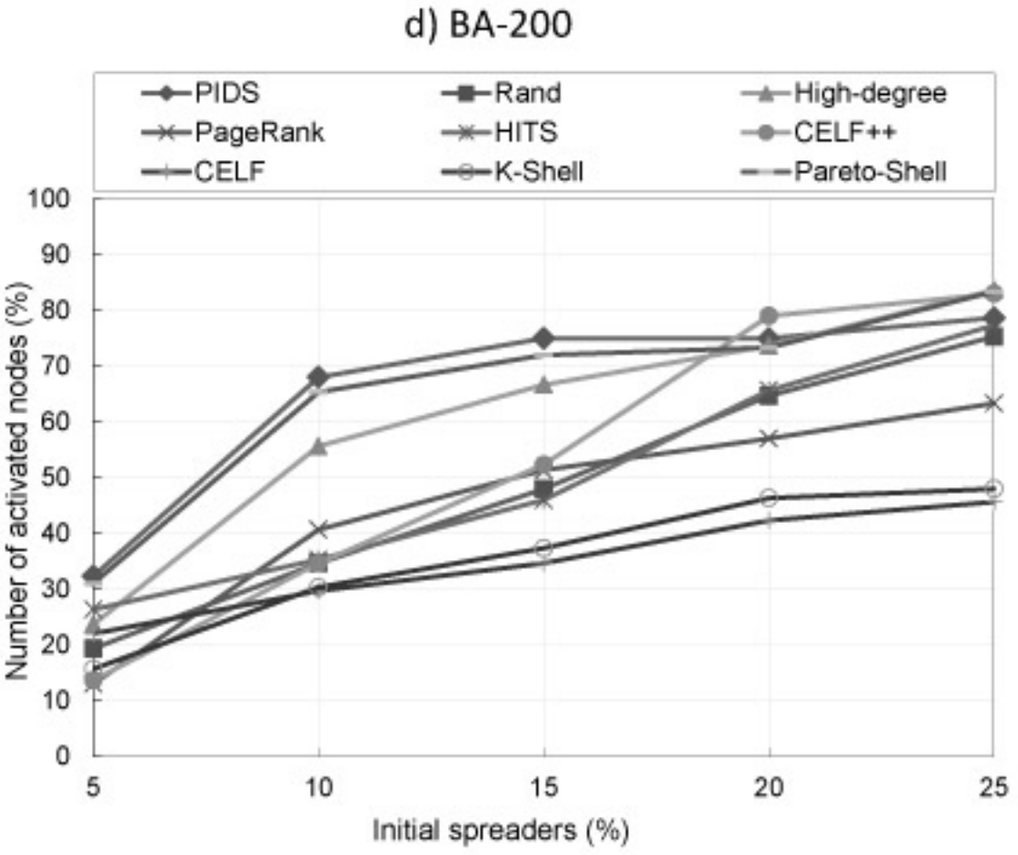
c) BA-150



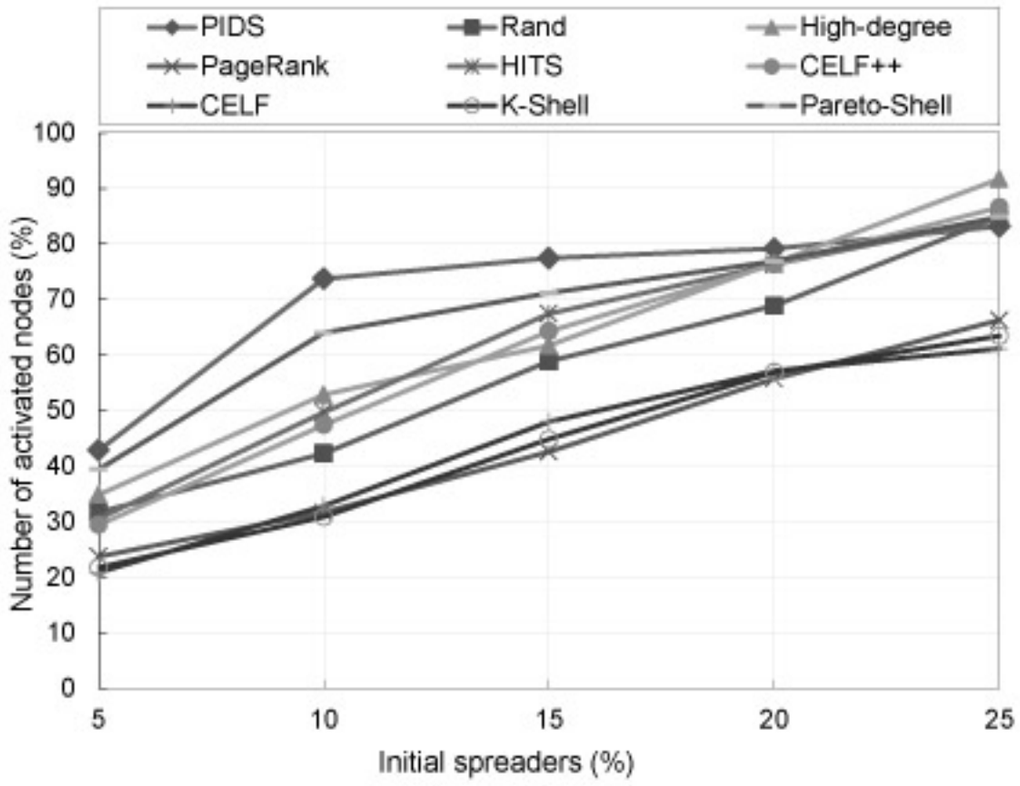
d) BA-200

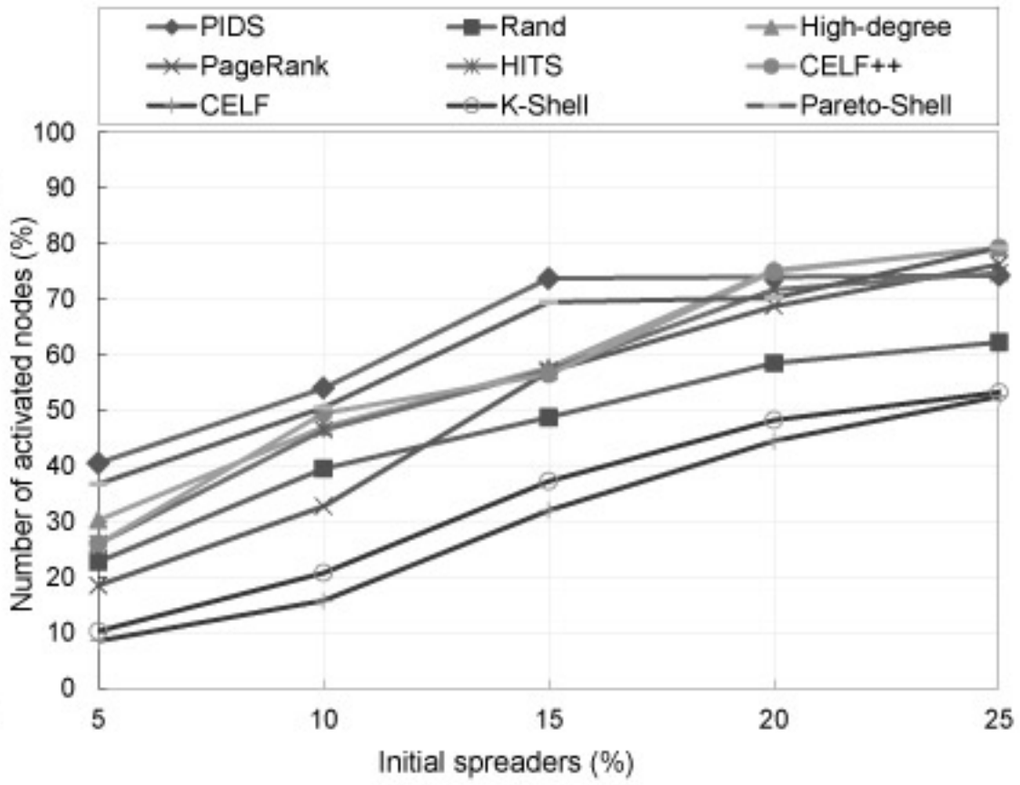


e) BA-250

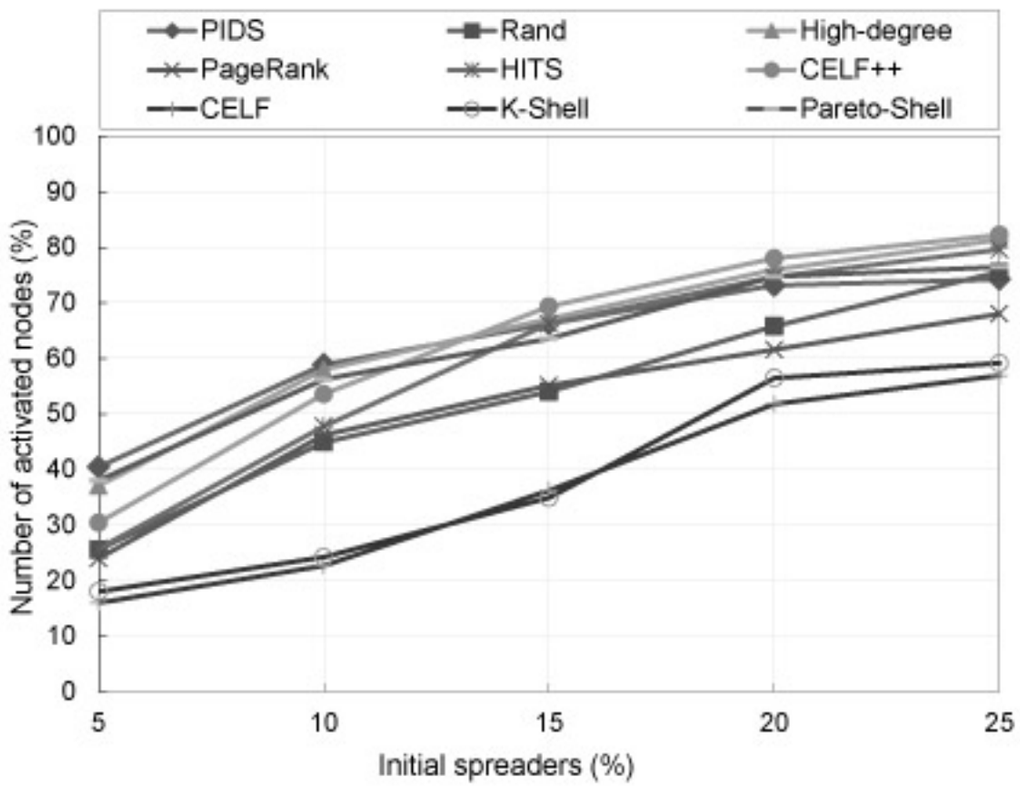


f) BA-300

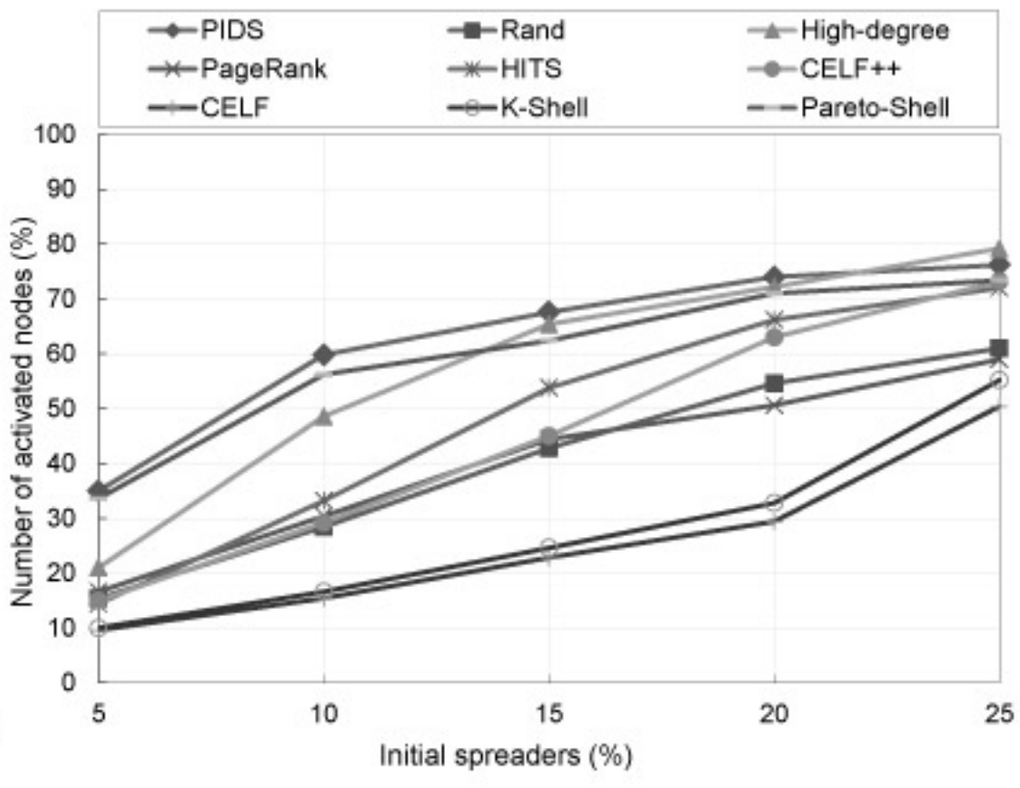




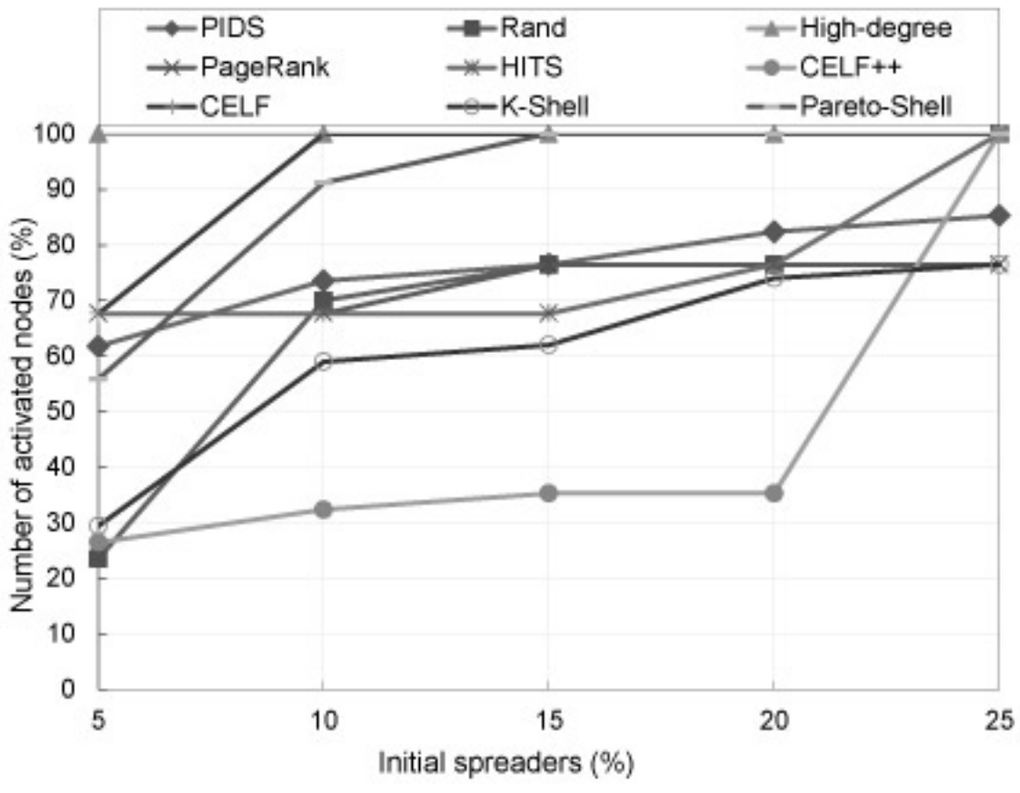
g) BA-350



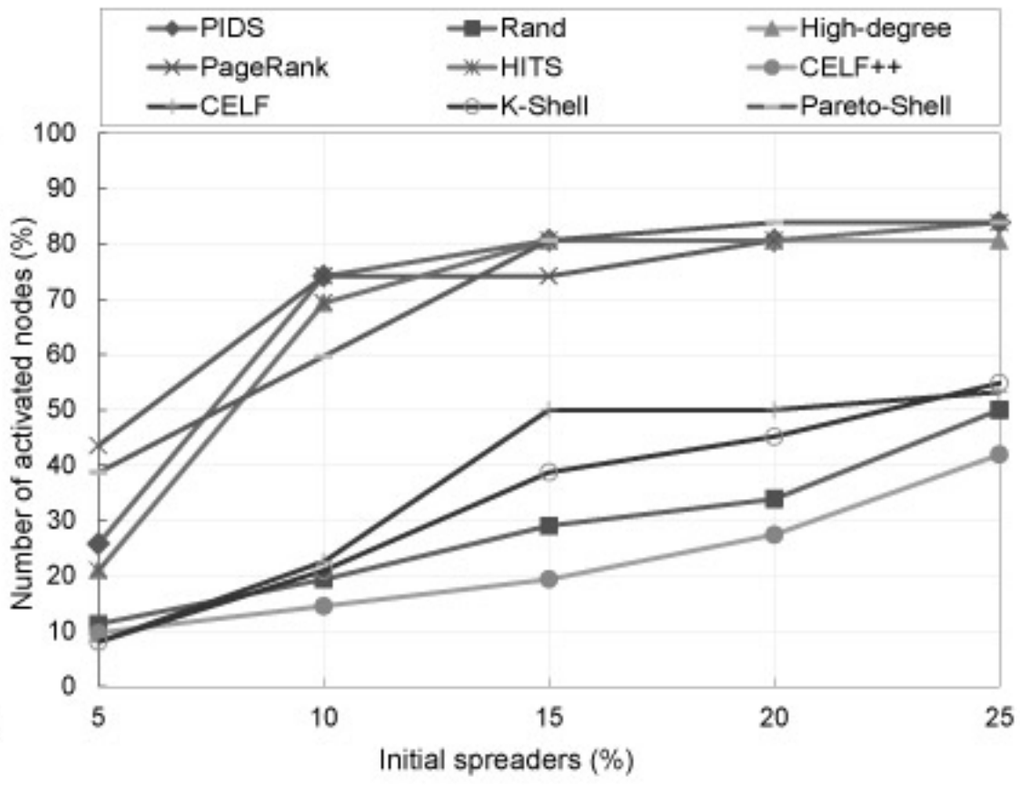
h) BA-400



i) BA-450



j) BA-500



k) Karate

l) Dolphins

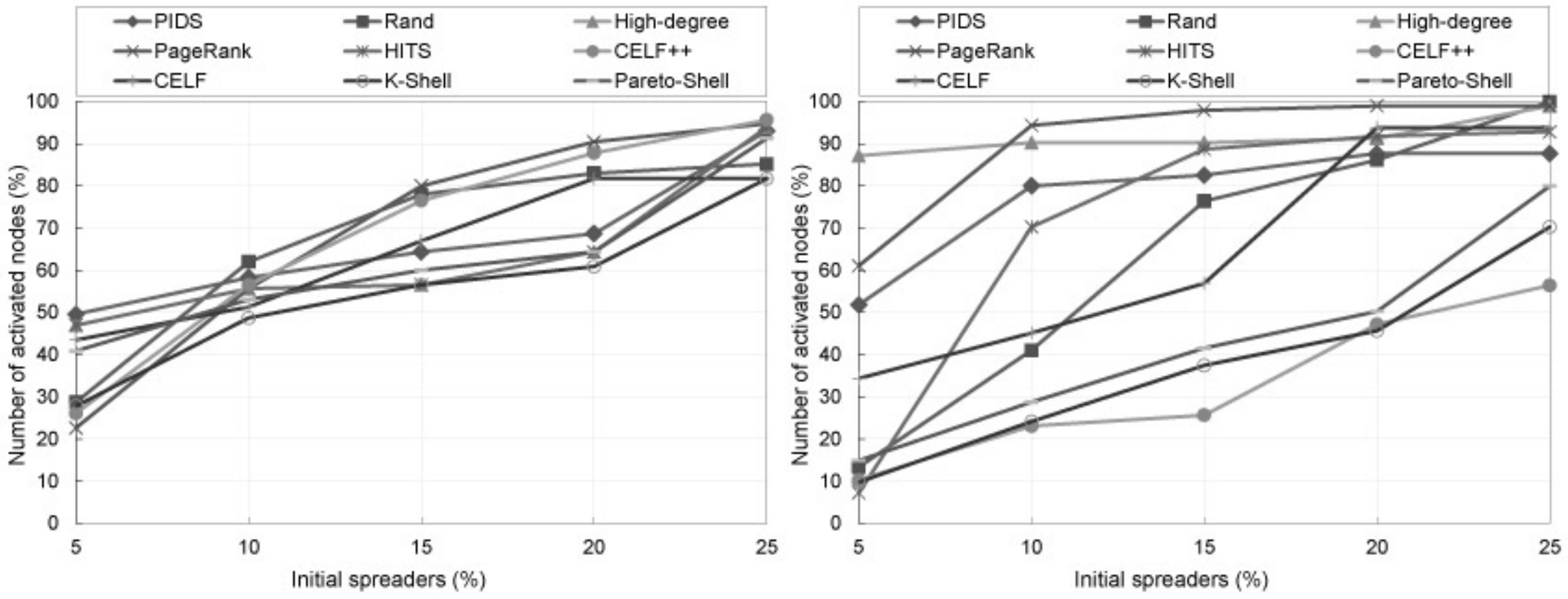


Figure 6. Numbers of activated nodes using different algorithms and initial numbers of activated nodes.

To verify statistical differences among the proposed algorithm (PIDS) and other algorithms, a two-tailed t-test for a confidence level of 95% has been performed to compare the algorithms from a statistical point of view (Table 5). In this test, it is assumed that the difference between each pair of algorithm is statistically significant if the difference significance is less than 0.05. The first column of table 5 includes the list of the algorithms for influence maximization. The second column shows the p-value of the t-test concerning PIDS versus each algorithm. In the third column, the t-test result concerning PIDS versus the corresponding algorithm is shown as '+', '–' and '~' when the performance of PIDS in terms of influence spread is significantly better than, worse than, and similar to that of the corresponding algorithm. This conclusion is drawn based on the t-test result for influence spread difference significance between the two compared algorithms. For example, in Table 5, for a comparison between PIDS and RAND, for 5% of initial spreaders, the p-value of 1.93e-25 indicates that PIDS outperforms Rand. From the results, one can conclude that for a small number of initial spreaders (5% through 15%), PIDS outperforms other algorithms significantly in terms of the influence spread, whereas for 20% of initial spreaders, PIDS outperforms for at least six out of the eight algorithms (except for high-degree) and has similar performance in comparison with Pareto-shell. For a large number of initial spreaders (25%), PIDS is only better than Rand, PageRank, CELF, CELF+ and Pareto-shell. This result might mean that the results of MPIDS are more appropriate for influence maximization with a small number of initial spreaders (i.e., real-life application with a limited budget) than with a small number of initial spreaders.

Table 5. Results of t-test for comparing the performance of PIDS versus other algorithms in terms of influence spread.

Algorithm	Numbers of activated nodes (%)									
	5		10		15		20		25	
	p-value	result	p-value	result	p-value	result	p-value	result	p-value	result

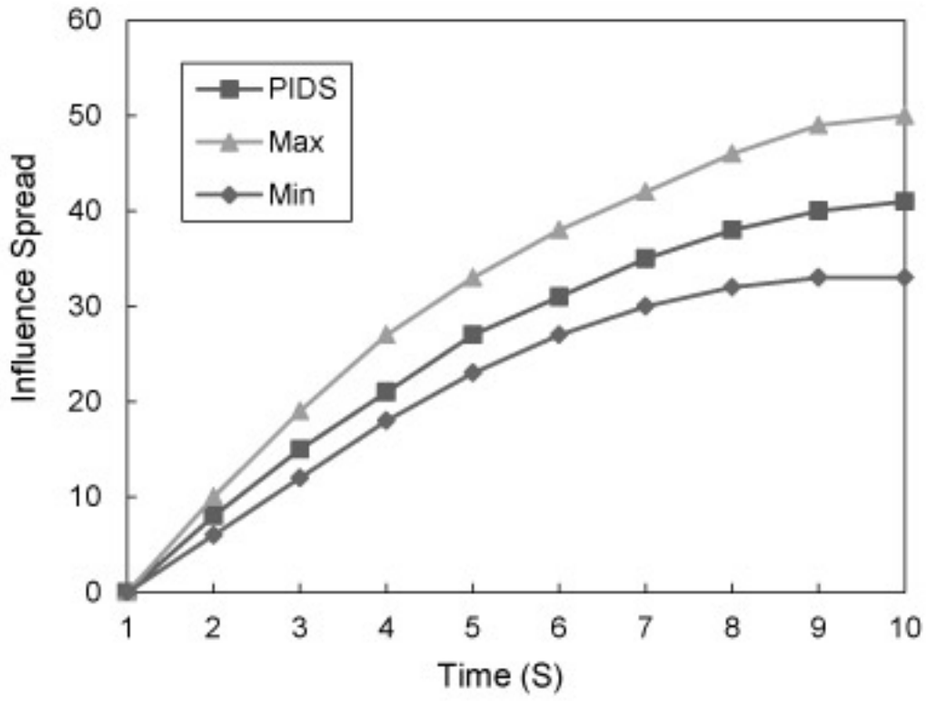
Rand	1.93e-25	+	9.77e-17	+	8.40e-15	+	2.51e-05	+	3.85e-02	+
High-degree	6.99e-03	+	6.07e-03	+	4.73e-03	+	9.76e-01	–	4.62e-04	–
PageRank	1.39e-11	+	1.30e-11	+	5.69e-09	+	4.58e-06	+	3.03e-07	+
HITS	1.37e-13	+	1.10e-11	+	7.64e-10	+	4.07e-02	+	2.45e-03	–
CELF	3.38e-23	+	3.90e-25	+	3.47e-21	+	8.18e-10	+	4.59e-02	+
CELF++	1.76e-17	+	8.43e-25	+	3.58e-22	+	1.43e-14	+	1.01e-16	+
K-Shell	1.16e-31	+	3.20e-40	+	1.69e-38	+	2.88e-35	+	8.02e-28	+
Pareto-Shell	3.07e-02	+	1.02e-03	+	1.83e-03	+	1.39e-01	~	2.41e-02	–

Note: '+', '–' and '~' indicate a 0.05 level of significance by t-test. '+', '–' and '~' denote that the performance of the PIDS is better than, worse than, and similar to that of the compared algorithm, respectively.

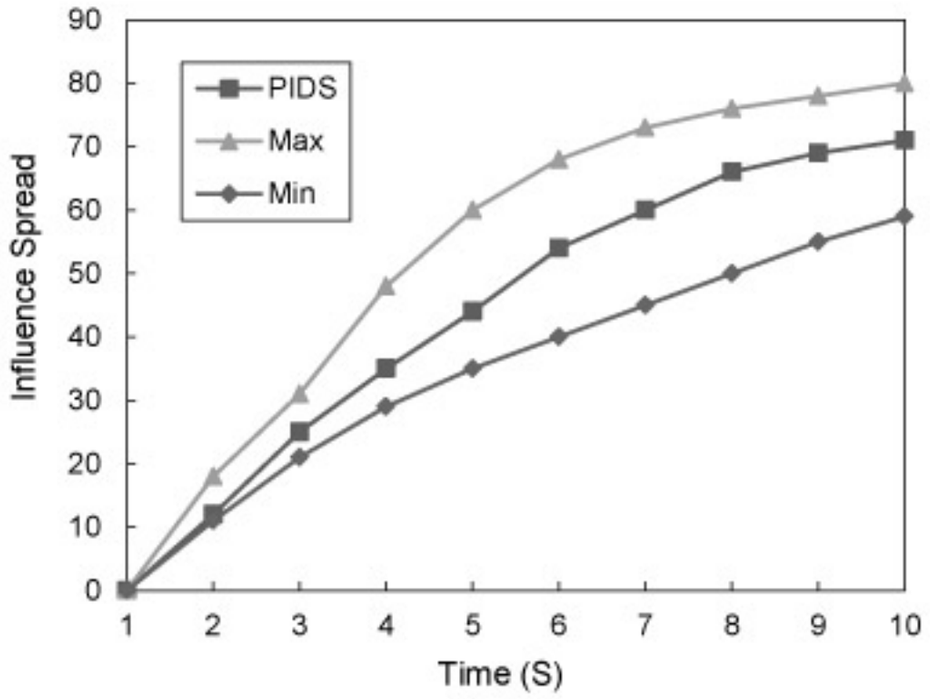
5.2.3 Convergence Behavior

In this experiment we aim to compare the convergence behavior of the proposed algorithm and other existing algorithms. In order to perform this experiment, we plot influence spread versus time in Figure 7 for each real and synthetic network. For clarity, we present only the curves for the best and the worst obtained results for the proposed algorithms. We note that in this experiment the best achievement of each algorithm occurred when 25% of the initial activated nodes are considered. According to the results in Figure 7, the following observations are noted:

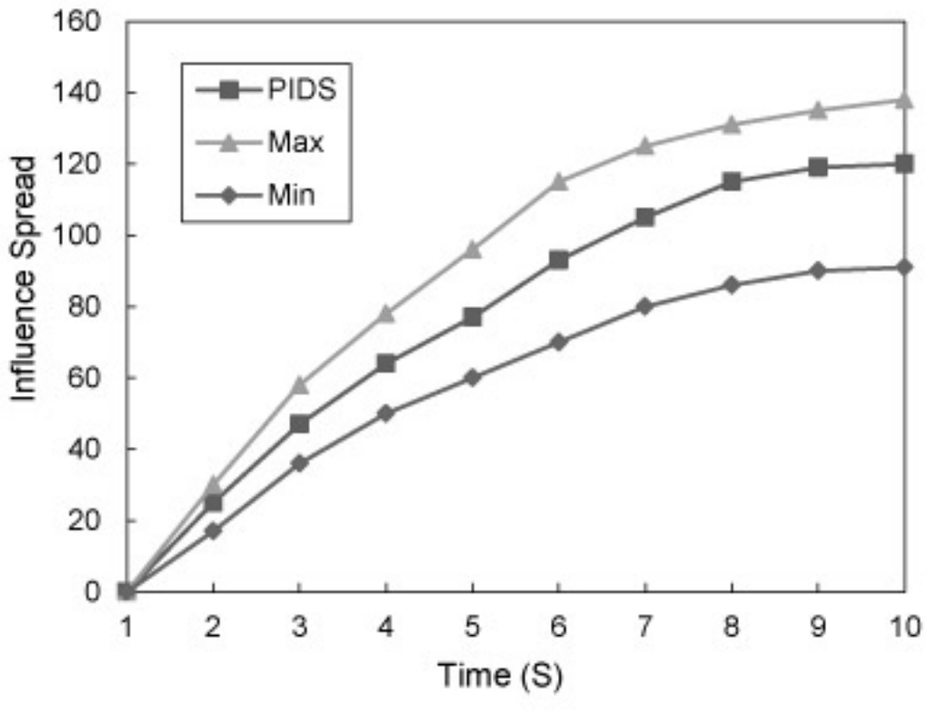
- In the initial time steps, the influence spread increases sharply; however, closer to convergence the trend becomes smoother.
- The proposed algorithm obtains greater spread influence in some of the cases including BA-100.
- In most cases, the Rand algorithm is superior to the others and the worst performance is obtained by CELF. The best and worst performances are shown by the maximum and the minimum curves in Figure 7, respectively.
- We note that in all cases, the proposed algorithm is very close to the best-obtained performance.



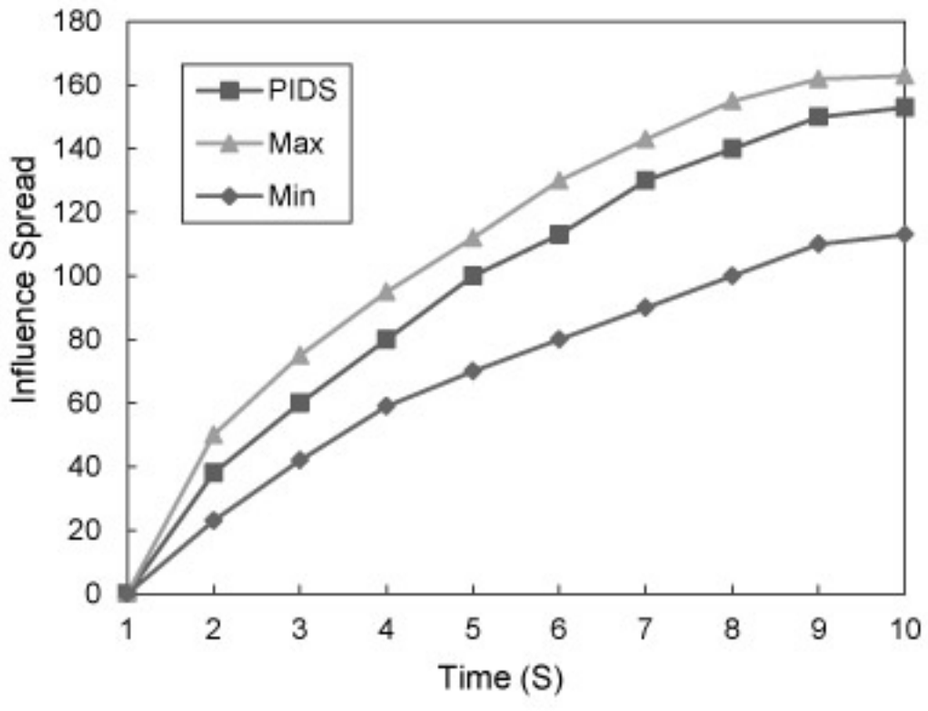
a) BA-50



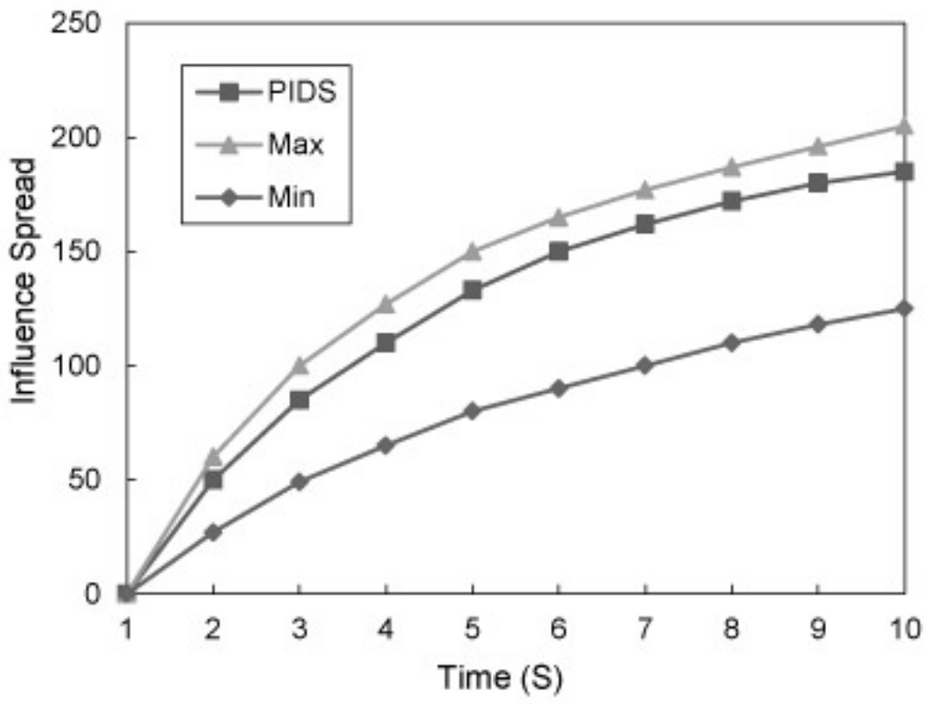
b) BA-100



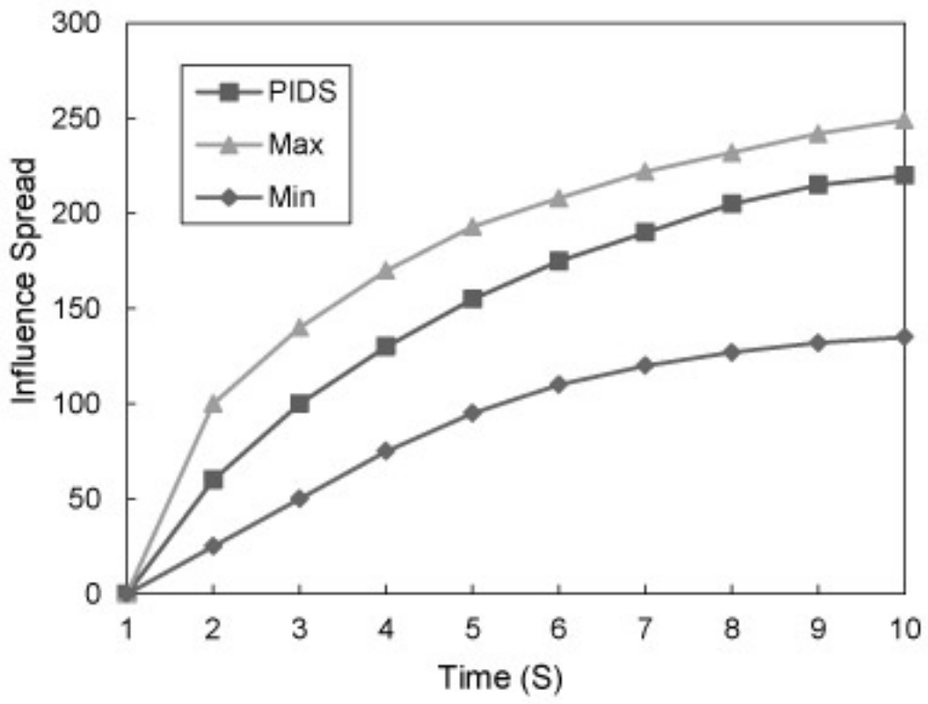
c) BA-150



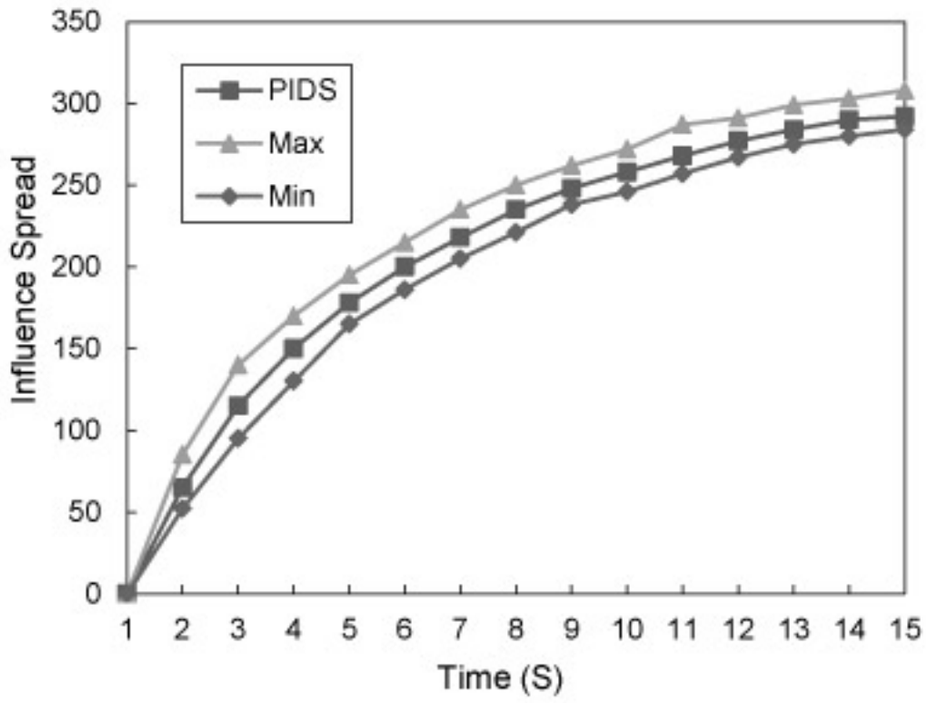
d) BA-200



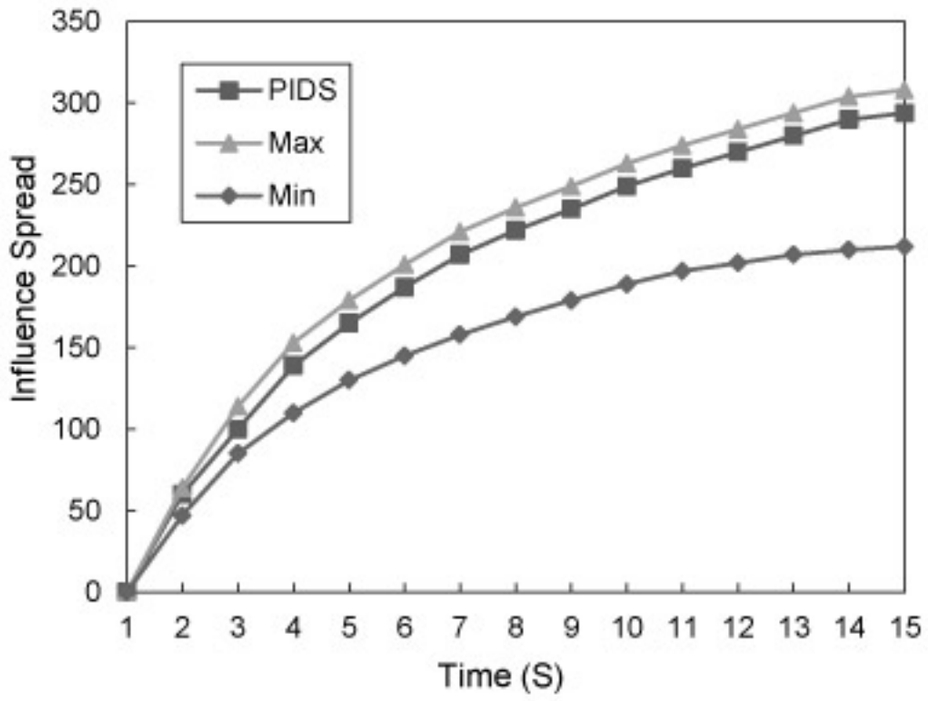
e) BA-250



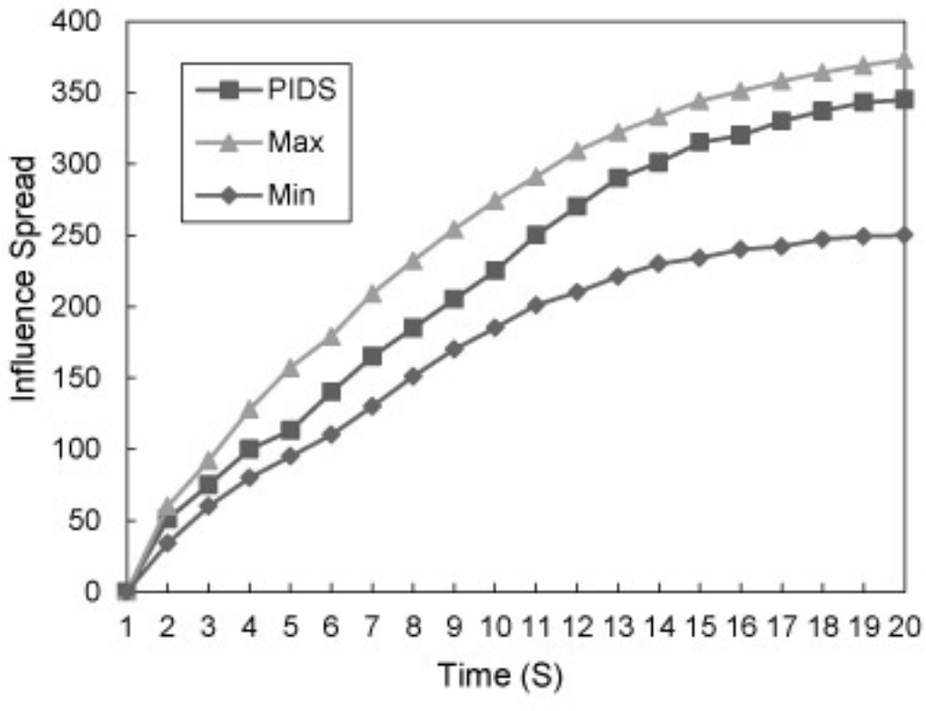
f) BA-300



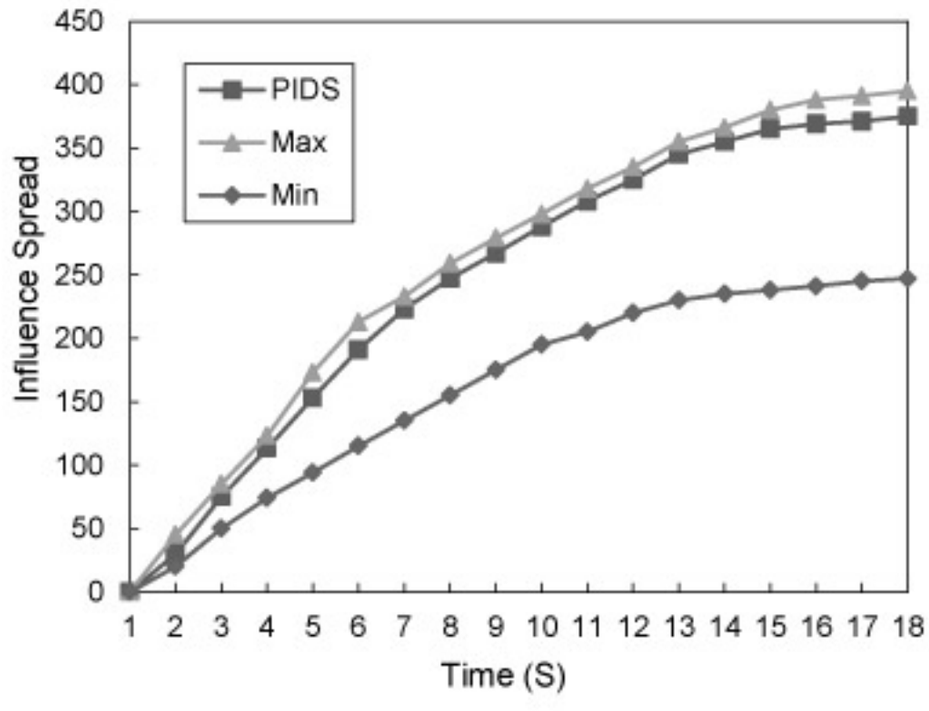
g) BA-350



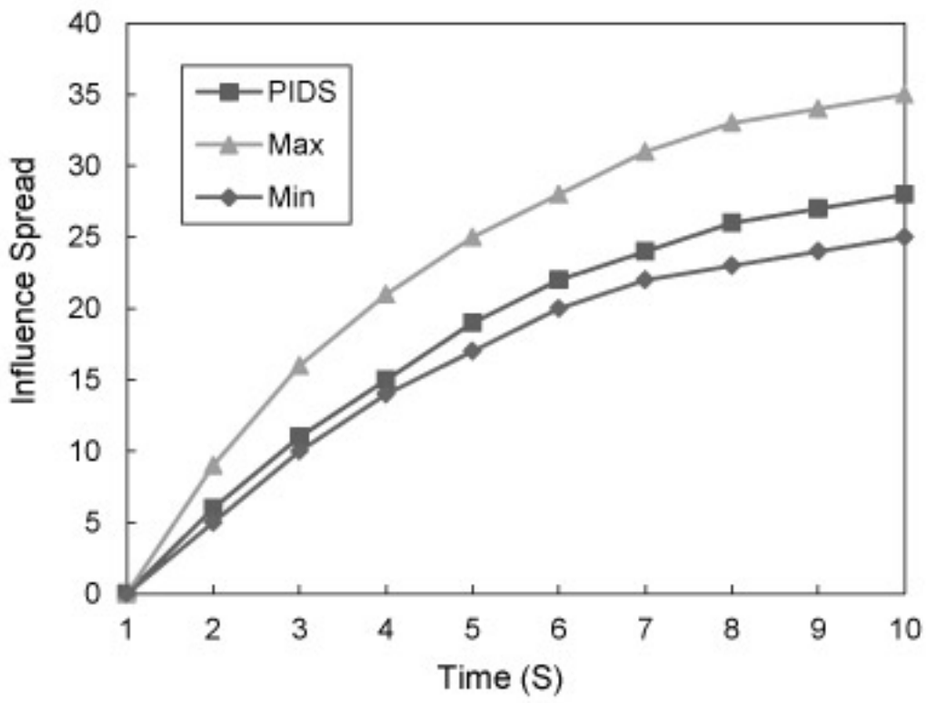
h) BA-400



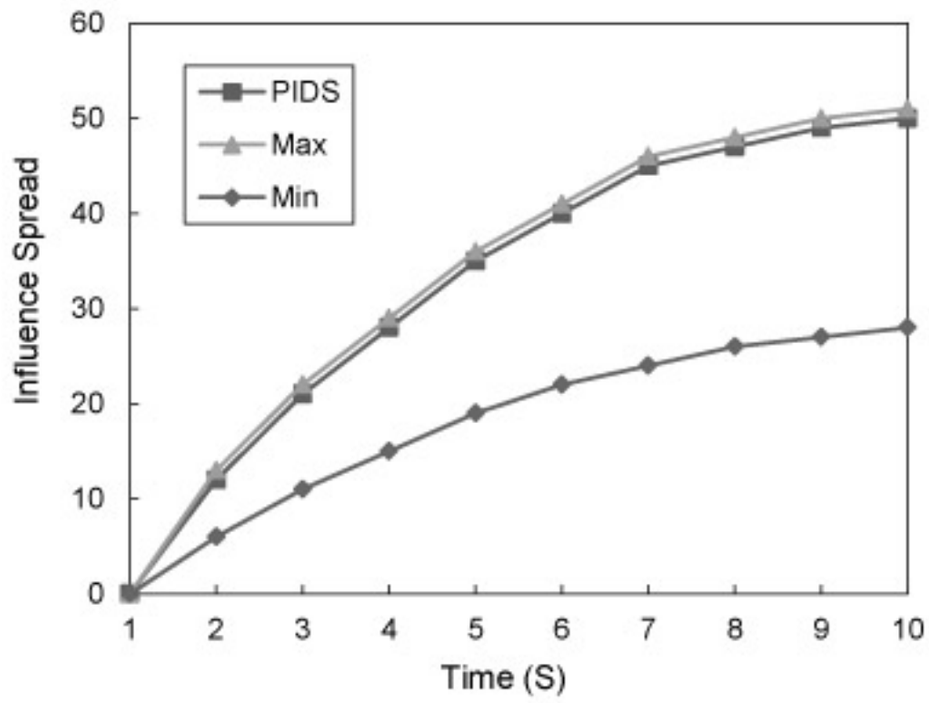
i) BA-450



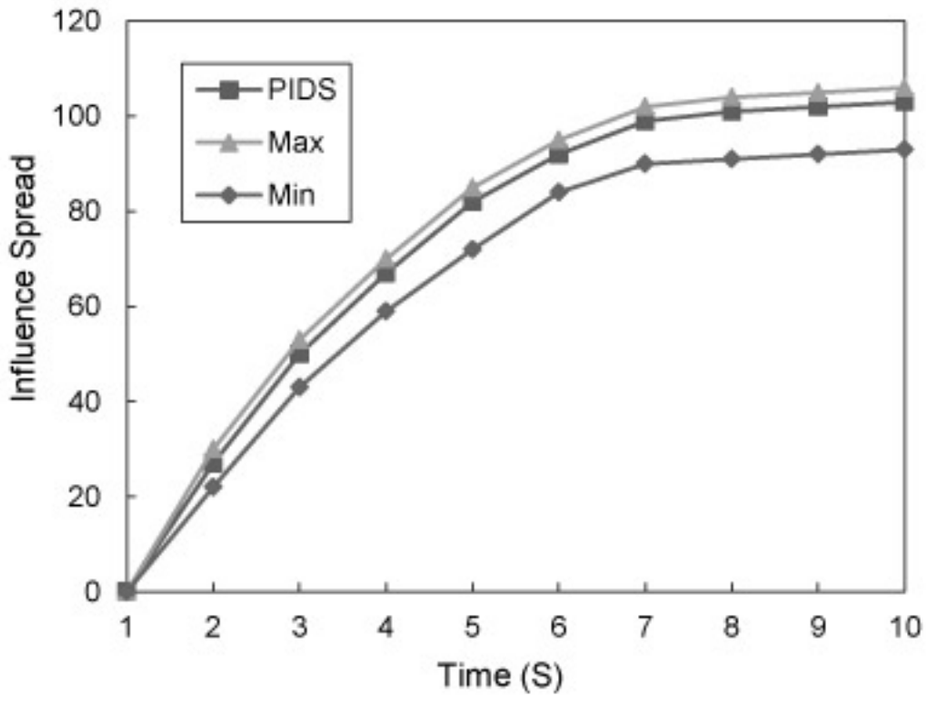
j) BA-500



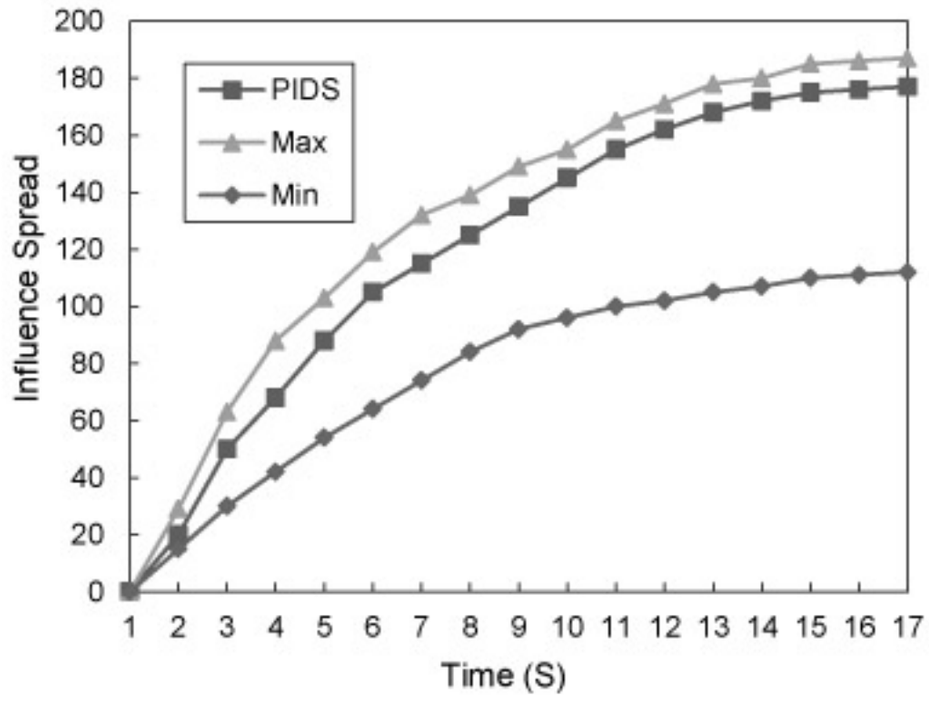
k) Karate



l) Dolphins



m) Football



n) Jazz

Figure 7. Convergence behavior of the proposed algorithm in terms of the influence spread.

6 Conclusion

In this paper, we propose a learning automaton-based algorithm for finding the minimum positive influence dominating set (MPIDS). The proposed algorithm iteratively finds candidate nodes in order to form a PIDS in a social network. In the proposed algorithm, each node has two possible choices that correspond to whether or not it belongs to the PIDS. In the process of the algorithm, each node learns how to select one of its actions in order to minimize the number of nodes selected per iteration. We then introduce the minimum positive influence dominating set in an application of influence maximization. We also discuss how to solve the influence maximization problem using the proposed learning automata-based algorithm and present the results of experiments performed to evaluate the performance of the proposed algorithm. These experimental results confirm the superiority of the proposed algorithm in terms of the total influence spread.

References

1. Watts DJ, Strogatz SH (1998) Collective dynamics of "small-world" networks. Nature 393:440–442.
2. Fortunato S (2010) Community detection in graphs. Physics Reports 486:75–174.
3. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. science 286:509–512.
4. 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. pp 137–146
5. Wang F, Camacho E, Xu K (2009) Positive influence dominating set in online social networks. In: Combinatorial Optimization and Applications. pp 313–321
6. Wang F, Du H, Camacho E, et al (2011) On positive influence dominating sets in social networks. Theoretical Computer Science 412:265–269.
7. Raei H, Yazdani N, Asadpour M (2012) A new algorithm for positive influence dominating set in social networks. In: Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012). IEEE, pp 253–257
8. Zhang W, Wu W, Wang F, Xu K (2012) Positive influence dominating sets in power-law graphs. Social Network Analysis and Mining 2:31–37.
9. Dinh TN, Shen Y, Nguyen DT, Thai MT (2014) On the approximability of positive influence dominating set in social networks. Journal of Combinatorial Optimization 27:487–503.
10. Wang G, Wang H, Tao X, Zhang J (2014) Finding weighted positive influence dominating set to make impact to negatives: a study on online social networks in the new millennium. In: ICTs and the Millennium Development Goals. Springer, pp 67–80
11. Java A, Kolari P, Finin T, Oates T (2006) Modeling the spread of influence on the blogosphere. In: Proceedings of the 15th international world wide web conference. pp 22–26
12. Kimura M, Saito K, Nakano R, Motoda H (2010) Extracting influential nodes on a social network for information diffusion. Data Mining and Knowledge Discovery 20:70–97.
13. Richardson M, Domingos P (2002) Mining knowledge-sharing sites for viral marketing. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp 61–70
14. 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
15. Liu B, Cong G, Zeng Y, et al (2014) Influence spreading path and its application to the time constrained social influence maximization problem and beyond. IEEE Transactions on Knowledge and Data Engineering 26:1904–1917.

16. Xu W, Lu Z, Wu W, Chen Z (2014) A novel approach to online social influence maximization. Social Network Analysis and Mining 4:1–13.
17. Lee J-R, Chung C-W (2015) A query approach for influence maximization on specific users in social networks. IEEE Transactions on knowledge and data engineering 27:340–353.
18. Brin S, Page L (1998) The anatomy of a large-scale hypertextual Web search engine. Computer networks and ISDN systems 30:107–117.
19. Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) 46:604–632.
20. Lü L, Zhou T, Zhang Q-M, Stanley HE (2016) The H-index of a network node and its relation to degree and coreness. Nature communications 7:10168.
21. 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
22. 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
23. Kelleher LL, Cozzens MB (1988) Dominating sets in social network graphs. Mathematical Social Sciences 16:267–279.
24. Molnár F, Sreenivasan S, Szymanski BK, Korniss G (2013) Minimum dominating sets in scale-free network ensembles. Scientific reports 3:1736.
25. Basuchowdhuri P, Majumder S (2014) Finding Influential Nodes in Social Networks Using Minimum k-Hop Dominating Set. In: International Conference on Applied Algorithms. Springer, pp 137–151
26. Kim D, Li D, Asgari O, et al (2013) A dominating set based approach to identify effective leader group of social network. In: International Computing and Combinatorics Conference. Springer, pp 841–848
27. Kanté MM, Limouzy V, Mary A, Nourine L (2011) Enumeration of minimal dominating sets and variants. In: International Symposium on Fundamentals of Computation Theory. Springer, pp 298–309
28. Torkestani JA, Meybodi MR (2012) Finding minimum weight connected dominating set in stochastic graph based on learning automata. Information Sciences 200:57–77.
29. Chvatal V (1979) A greedy heuristic for the set-covering problem. Mathematics of operations research 4:233–235.
30. Johnson DS (1974) Approximation algorithms for combinatorial problems. Journal of computer and system sciences 9:256–278.
31. Slavík P (1996) A tight analysis of the greedy algorithm for set cover. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. ACM, pp 435–441
32. Nieberg T, Hurink J (2005) A PTAS for the minimum dominating set problem in unit disk graphs. In: International Workshop on Approximation and Online Algorithms. Springer, pp 296–306
33. Nacher JC, Akutsu T (2013) Analysis on critical nodes in controlling complex networks using dominating sets. In: 2013 International Conference on Signal-Image Technology & Internet-Based Systems (SITIS),. IEEE, pp 649–654
34. Narendra KS, Thathachar MA (1989) Learning automata: an introduction. Prentice-Hall
35. Damerchilu B, Norouzzadeh MS, Meybodi MR (2016) Motion estimation using learning automata. Machine Vision and Applications 27:1047–1061.
36. Mofrad MH, Sadeghi S, Rezvanian A, Meybodi MR (2015) Cellular edge detection: Combining cellular automata and cellular learning automata. AEU-International Journal of Electronics and Communications 69:1282–1290.
37. Rezvanian A, Meybodi MR (2015) Finding Minimum Vertex Covering in Stochastic Graphs: A Learning Automata Approach. Cybernetics and Systems 46:698–727.
38. Kordestani JK, Ahmadi A, Meybodi MR (2014) An improved Differential Evolution algorithm using learning automata and population topologies. Applied Intelligence 41:1150–1169.

39. Mahdaviani M, Kordestani JK, Rezvanian A, Meybodi MR (2015) LADE: Learning Automata Based Differential Evolution. International Journal on Artificial Intelligence Tools 24:1550023.
40. Meybodi MRM, Meybodi MR (2014) Extended distributed learning automata. Applied Intelligence 41:923–940.
41. Rezvanian A, Meybodi MR (2016) Stochastic Graph as a Model for Social Networks. Computers in Human Behavior 64:621–640.
42. Rezvanian A, Meybodi MR (2015) Finding maximum clique in stochastic graphs using distributed learning automata. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 23:1–31.
43. Esnaashari M, Meybodi MR (2013) Deployment of a Mobile Wireless Sensor Network with k-Coverage Constraint: A Cellular Learning Automata Approach. Wireless Networks 19:945–968.
44. Elyasi M, Meybodi M, Rezvanian A, Haeri MA (2016) A fast algorithm for overlapping community detection. In: 2016 Eighth International Conference on Information and Knowledge Technology (IKT). 2016, pp 221–226
45. Khomami MMD, Rezvanian A, Meybodi MR (2016) Distributed learning automata-based algorithm for community detection in complex networks. International Journal of Modern Physics B 30:1650042.
46. Rezvanian A, Meybodi MR (2017) A new learning automata-based sampling algorithm for social networks. International Journal of Communication Systems 30:e3091.
47. Rezvanian A, Meybodi MR (2017) Sampling algorithms for stochastic graphs: a learning automata approach. Knowledge-Based Systems in-press:1–19.
48. Mofrad MH, Jalilian O, Rezvanian A, Meybodi MR (2016) Service level agreement based adaptive Grid superscheduling. Future Generation Computer Systems 55:62–73.
49. Morshedlou H, Meybodi MR (2014) Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments. IEEE Transactions on Cloud Computing 2:156–167.
50. Vahidipour SM, Meybodi MR, Esnaashari M (2017) Adaptive Petri net based on irregular cellular learning automata with an application to vertex coloring problem. Applied Intelligence 46:272–284.
51. Masoumi B, Meybodi MR (2011) Speeding up learning automata based multi agent systems using the concepts of stigmergy and entropy. Expert Systems with Applications 38:8105–8118.
52. Brandes U (2001) A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25:163–177.
53. Freeman LC (1979) Centrality in social networks conceptual clarification. Social networks 1:215–239.
54. Torkestani JA, Meybodi MR (2011) A link stability-based multicast routing protocol for wireless mobile ad hoc networks. Journal of Network and Computer Applications 34:1429–1440.
55. Lakshmivarahan S, Thathachar MAL (1976) Bounds on the convergence probabilities of learning automata. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 6:756–763.
56. Zachary WW (1977) An information flow model for conflict and fission in small groups. Journal of anthropological research 452–473.
57. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99:7821–7826.
58. Lusseau D, Schneider K, Boisseau OJ, et al (2003) The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. Behavioral Ecology and Sociobiology 54:396–405.
59. Gleiser PM, Danon L (2003) Community structure in jazz. Advances in complex systems 6:565–573.
60. Zeng A, Zhang C-J (2013) Ranking spreaders by decomposing complex networks. Physics Letters A 377:1031–1035.
61. Yeruva S, Devi T, Reddy YS (2016) Selection of influential spreaders in complex networks using Pareto Shell decomposition. Physica A: Statistical Mechanics and its Applications 452:133–144.
62. Toothaker LE (1993) Multiple comparison procedures. Sage