Applied Soft Computing Journal 82 (2019) 105554

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

LAPSO-IM: A learning-based influence maximization approach for social networks

Shashank Sheshar Singh [*], Ajay Kumar, Kuldeep Singh, Bhaskar Biswas

Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, 221–005, India

HIGHLIGHTS

- We present a *learning automata based discrete particle swarm optimization* (LAPSO-IM) algorithm for influence maximization.
- We extend local influence evaluation function *expected diffusion value* (EDV) to approximate within the two-hope area.
- Classical Linear Threshold and Cascade diffusion models are utilized.
- The proposed algorithm is a trade-off between quality and efficiency.
- The statistical tests are performed to show the significant difference between the proposed algorithm and the state-of-the-art algorithms.

ARTICLE INFO

Article history:
Received 16 August 2018
Received in revised form 5 April 2019
Accepted 30 May 2019
Available online 7 June 2019

Keywords:
Influence maximization
Information diffusion
Social networks
Particle swarm optimization
Learning automata

ABSTRACT

Online social networks play a pivotal role in the propagation of information and influence as in the form of word-of-mouth spreading. *Influence maximization* (IM) is a fundamental problem to identify a small set of individuals, which have maximal influence spread in the social network. IM problem is unfortunately NP-hard. It has been depicted that hill-climbing greedy approach gives a good approximation guarantee. However, it is inefficient to run a greedy approach on large-scale social networks. In this paper, a local influence evaluation function is presented for optimizing IM problem. The local influence evaluation function provides a reliable expected diffusion value of influence spread under the linear threshold, independent and weighted cascade models. To optimize local influence evaluation function, a *learning automata based discrete particle swarm optimization* (LAPSO-IM) algorithm is proposed. LAPSO-IM redefines the update rule of particle's velocity based on *learning automata* action to overcome the weakness of premature convergence. The experimental results on six real-world social networks show that the proposed algorithm is more effective than base algorithm DPSO with same the level of efficiency and more time-efficient than IMLA with approximate influence spread.

1. Introduction

Nowadays, the popularity and growth of online social networks such as Twitter, Facebook, LinkedIn, and microblogs, etc., have brought the attention of researchers towards social networks analysis. Beside user's interaction and communication, social networks also provide a platform to companies for marketing and advertising of products. Social networks propagate information about products via links between users using word-of-mouth effect. This word-of-mouth advertising has immense application potential in viral marketing [1,2], rumor control [3,4], social recommendation [5], revenue maximization [6], and network monitoring [7], etc. In viral marketing, the advertising companies give free samples of the product to a group of users to maximize product adoption using the word-of-mouth effect.

Let us recall the example of the IM problem in viral marketing presented by Kempe et al. [8]. Given a social network with the influence weight which estimates the extent of influence to one another, and the social network provides a medium of interaction for advertising and marketing. A marketing company wants to advertise a new product with the hope of the maximal adoption of the product. The company targets a small subset of influential users by giving them a free sample of the product, by considering the cost of the product and limited budget. The company hopes that targeted individuals will motivate others to adopt the product with word-of-mouth influence propagation and maximize the influence spread. Therefore, the viral marketing strategy introduced the *influence maximization* (IM) problem. Inspired by the idea of viral marketing, Pedro and Matt [2] were first to

* Corresponding author.
E-mail addresses: shashankss.rs.cse16@iitbhu.ac.in (S.S. Singh), ajayk.rs.cse16@iitbhu.ac.in (A. Kumar), kuldeep.rs.cse13@iitbhu.ac.in (K. Singh), bhaskar.cse@iitbhu.ac.in (B. Biswas).

https://doi.org/10.1016/j.asoc.2019.105554
1568-4946/© 2019 Elsevier B.V. All rights reserved.

introduced IM as an optimization problem. Formally, IM problem is formatted by Kempe et al. [8], given as follows.

$$S^* = \underset{|S|=k}{argsmax}\{\sigma(S)\} \tag{1}$$

where $\sigma(S)$ is an objective function which gives the expected influence spread in the network after completion of the diffusion process. Kempe et al. proved that the objective function $\sigma(S)$ is sub-modular. The authors pointed out that IM is NP-hard under traditional diffusion models (*linear threshold* (LT), *independent cascade* (IC) and *weighted cascade* (WC) models). To solve IM problem, they introduced a greedy algorithm and proved that the greedy solution is approximated to within a factor of $(1-1/e-\varepsilon)$.

All the existing approaches have their own weaknesses. Simulation-based approaches [9–13] rely on time-consuming *Monte-Carlo* (MC) simulations so that these approaches are computationally inefficient. Proxy-based approaches [14–18] compromise some precision in influence spread to improve computational efficiency. Sketch-based approaches [19–21] are theoretical efficient but these approaches are aligned with some specific diffusion models.

Contribution. Discrete particle swarm optimization (DPSO) [22] is one of the most popular method for optimizing NP-hard problems due to its simplicity and efficiency. The major weakness of DPSO is its premature convergence which leads to trapping in local optima. To maintain a trade-off between local and global search process, Hasanzadeh et al. [23] proposed a *learning automata* (LA) based DPSO named as DPSOLA. In this paper, we present a *learning automata based discrete particle swarm optimization* (LAPSO-IM) for IM problem in the social networks. The proposed algorithm adopted LA to utilizes the flexible self-adoption and automatic learning capability to learn the budget allocation for each iteration. LA is effective in LAPSO-IM because it controls the velocity of the particles and regulates particles movement during the search process which enables particles to dynamically search local and global space. Therefore, particles do not lose their useful search information and particle's variation will be increased which increases the probability of escaping from local minima. Hence, the use of LA in the proposed algorithm avoids the premature convergence and produces more effective seed set to maximize influence spread. The major contributions of our work are as follow.

- We extend local influence evaluation function *expected diffusion value* (EDV) [24] to approximate within the two-hope area as $EDV^{(2)}(S)$.
- We present a learning automata based PSO approach to optimize local influence evaluation function by redefining the velocity and position rules based on learning automata.
- We experimentally compare the proposed LAPSO-IM with the state-of-the-art algorithms in terms of quality and efficiency under traditional diffusion models.
- The experimental results on six real-world networks show that the proposed algorithm LAPSO-IM performs better than the base method DPSO in terms of influence spread with almost same running time. LAPSO-IM is more time-efficient than base method IMLA with the same level of influence spread.

Paper organization. Section 2 presents related work of influence maximization. Section 3 defines the problem statement and explains background information for better understanding of concepts. Section 4 describes the proposed work. Section 5 explains the algorithm LAPSO-IM with a running example. Section 6 discusses the experimental setup and result analysis of LAPSO-IM on real-world datasets. Finally, Section 7 is devoted to our conclusion and directions for future work.

2. Related work

Pedro and Matt [2] were first to introduced IM problem. Formally, IM problem is formatted by Kempe et al. [8]. Svirdenko et al. [25] extend the IM problem defined by [8] considering non-uniform selection cost. Suppose, each node u has a seed selection cost $c_s(u)$, then Knapsack greedy aims to select k seed nodes S bounded by budget B, i.e., $S \leftarrow argmax_{S^* \subseteq V \wedge |S^*|=k \wedge c_s(S^*) \leq B} \sigma(S^*)$. The objective function $\sigma(S^*)$ is still sub-modular under traditional diffusion models. Therefore, hill-climbing greedy can be applied with knapsack constraint.

In recent years, many studies have been proposed to enhance the efficiency of the greedy algorithm. Using *diminishing returns* property of a sub-modular function, Leskovec et al. [9] introduced an approach viz. *cost-effective lazy forward* (CELF). It uses hill-climbing greedy approach to find seed nodes. It stores marginal gain of each node with the current seed set. In each iteration, it reduces search space significantly based on *diminishing returns* property. Therefore, CELF is up to 700 times faster than greedy with the same level of influence spread. Inspired by CELF, Goyal et al. [12] proposed CELF++ algorithm. In each iteration, it computes marginal gain of each node with the current seed set S as well as $S \cup \{v\}$, where v represent the highest marginal gain node by now. It avoids re-computation of marginal gain for non-candidate seed nodes. Thus, CELF++ is 30–50% faster than CELF. Ge et al. [26] proposed a learning automata based approach named as *learning automata based influence maximization* (LAIM). It is a self-adaptive decisive approach to exploit *sub-modular* property. It uses the greedy hill-climbing technique to find seed set. It requires lesser number of MC simulation. Therefore, LAIM is more time-efficient than CELF.

Some heuristic approaches, including degree and other centrality based methods [15,27], influence path based methods [14,16,28,29], score estimation based methods [17,30,31], and meta-heuristic methods [32,33] select the top-k influential nodes according to the property of each single node. Chen et al. [15] proposed a degree-based heuristic viz. *degree discount* (DD). The incoming neighbors $v \in N_{inc}(u)$ of a node u are not able to influence u, when u is selected as seed. The algorithm iteratively selects the highest degree node followed by DD step. In degree discount step, it reduces the degree of each node v by one. Therefore, DD performs better than the highest degree. Although, improvement of DD is very limited. Kundu and Pal [27] proposed a deprecation based greedy strategy by estimating the performance of nodes and marking the nodes to be deprecated. They have proved that the method can provide a guaranteed target node set when the influence function is monotonic and sub-modular. Kimura et al. [14] introduced an influence path based approach viz. *shortest path 1 model* (SP1M). It considers only the shortest and second shortest paths to propagate influence in the network. Therefore, influence spread can be calculated recursively. To improve its performance, an approximation strategy is used to define objective function. Inspired by SP1M, Chen et al. [16] proposed an algorithm named *maximum influence arborescence* (MIA). To estimate influence spread, it maintains local arborescence structure. MIA considers only highest propagation probability paths to evaluate influence spread. Therefore, it is more time efficient than SP1M.

Some sketch-based approaches [19–21,34] are designed to improve the theoretical efficiency of simulation-based approaches. Cheng et al. [19] proposed an algorithm viz. StaticGreedy which guarantees the sub-modularity of objective function $\sigma(S)$ in seed selection process. This algorithm uses snapshot-based sampling. Therefore, its time complexity is reduced fundamentally without compromising on accuracy. Ohsaka et al. [20] introduced snapshot-based sampling algorithm viz. PRUNEDMC. To reduce the cast of MC simulations, it updates the outputs of MC simulations iteratively. It also prunes the breadth-first search to avoid re-computation. It outperforms the state-of-the-art approaches in terms of influence spread. Tang et al. [21] proposed an approach *two-phase influence maximization* (TIM) based on reverse-reachability sampling. It uses a novel heuristic to enhance its practical efficiency without compromising on asymptotic efficiency. Recent years, context-aware IM problems extend the generic IM problem by considering contextual features of the item. These IM problems consider contextual features such as temporal [35], topical [18,36], competitive [37], and spatial information [38] to improve effectiveness of seed.

As we discussed earlier, the three tribes of IM approaches suffer from their own flaws respectively. Simulation-based approaches are computationally intensive due to a large number of MC simulations, heuristic approaches pursue a high efficiency by sacrificing some degree of accuracy, and sketch-based approaches is just a compromise between fast convergence and high accuracy. The objective function $\sigma(.)$ is a random function, therefore it can be perceived as a random environment in terms of LA. In consideration of LA's robustness and fast convergence in random environments, it is a natural idea to apply the fruitful achievements in the field of LA to solving IM problem. Hasanzadeh et al. [23] introduced a LA based particle swarm optimization approach to avoid local optima with better convergence. There are some works are presented for noisy optimization, such as LAPSO [39], and LANBC [40]. In this paper, we adopt DPSOLA [23] to optimize the IM problem.

2.1. Business implications of IM problem

In recent years, the importance of micro-blog services as a marketing tool has multiplied, and its usage has spanned into diverse areas. For example, suppose that we are trying to market a product, an idea, innovation or behavior within a population of individuals, the commonly used advertising channel for such use case is peddling, putting signboards or playing promotion music. A promising way that could be used for such promotions is to place ads (advertisements) on popular micro-blog services (e.g., Twitter) through the viral marketing channel. Especially, if a company wants to promote its newly produced camera by placing ads, the company can convince several key people in the social network (influencer's) to adopt or try the new camera first, then the company may utilize the diffusion effect over the network to increase the effectiveness of its marketing campaign. If we assume that convincing each key person to spread the news on the new camera costs money, then a very significant problem could be described as: Given a social network, how can we identify the key people through which we can spread the promotions for the new product most efficiently?

Recent empirical advances have used new observational techniques as well as randomized experiments to identify influence and susceptibility in networks. These advances provide new opportunities for specifying more accurate, contextual influence models when using influence maximization to identify optimal targets of public policy interventions or business advertising. Our results suggest that the growing body of research on influence maximization needs to incorporate results and insight from the empirical literature on influence identification to become more realistic and practically applicable. There are also some other business implications of IM problem like revenue maximization, and profit maximization from the perspective of advertiser and network service provider both. Apart from business advertising, there are some other potential applications of IM problem such as political campaign or elections, trend analysis and sales predictions, network monitoring, counter terrorism efforts, epidemiology, contagion management, etc.

Table 1
Notations.

$G(V, E)$	$\triangleq$	A social network with vertex set V and edge set E
$N(u)$	$\triangleq$	The neighbors set of node u
S	$\triangleq$	Seed set
k	$\triangleq$	The number of nodes in seed set ($\lvert S\rvert$)
$\sigma(S)$	$\triangleq$	The expected influence of seed set in the network
pop	$\triangleq$	The population of swarm
$V(j)$	$\triangleq$	The velocity vector of particle j
$pos(j)$	$\triangleq$	The position vector of particle j
$pbest(j)$	$\triangleq$	The personal best position vector of particle j
$gbest$	$\triangleq$	The global best position vector (solution)
I_{max}	$\triangleq$	The maximum number of iteration
δ	$\triangleq$	Inertia weight $\delta \in [0, 1]$

To conclude, the phenomenon of viral marketing has made enterprises sit up and take notice. Businesses are now including viral marketing techniques into their customer relationship management efforts to communicate with their existing and prospective customers. As a businessperson, it depends on your strategic efforts in creating a viral marketing strategy that will have a lasting positive influence on your business.

3. Preliminaries

3.1. Notations and definitions

Notations that are needed for problem formulations in this paper are given in Table 1. Some definitions are given related to research work as follow.

Definition 1 (*Influence Graph*)**.** An influence graph is a directed graph $G(V, E)$ that shows the relationship between users. Here, V and E represent a set of users and a set of edges respectively.

Definition 2 (*Neighbors*)**.** Neighbors $N(u)$ of node u is defined as the set of users v such that $v \in N(u)$ iff $\exists(u, v) \in E$, $v \in V$. $N_{inc}(u)$ and $N_{out}(u)$ denote in and out neighbors of node u respectively.

Definition 3 (*Degree Centrality*)**.** Degree centrality is defined as the number of links incident upon a node i.e. $C_D(u) = |N(u)|$.

Definition 4 (*Seed Nodes*)**.** Seed nodes (S) are the set of nodes who act as the source of the information propagation process in the social network, $|S| = k$, $S \in V$.

Definition 5 (*Active Node*)**.** A node $u \in V$ is called active if either $u \in S$ or u adopted the information propagated by previously active nodes $v \in V_A$ under diffusion model M. Once u is activated, then $V_a \leftarrow \{V_A \cup u\}$.

Definition 6 (*Influence Spread*)**.** Influence spread $I_S(S)$ of the seed set S is defined as the number of active users after diffusion process under a diffusion model M, i.e., $I_S(S) = |V_A(S)|$.

3.2. Diffusion model

Kempe et al. [8] incorporated two basic diffusion models, *linear threshold* (LT) and *independent cascade* (IC) for information propagation. In both of these models, each node at any time-stamp t_s belongs to one of the two state: inactive and active. Nodes that are not influenced by their neighbors or not heard about the product are known as inactive nodes. Initially at time ($t_s = 0$), all nodes are inactive. Active nodes are influenced by their neighbors and only such nodes can propagate influence to their neighbors. There is a survey on traditional diffusion model [41].

3.2.1. Linear threshold model

In this model, every node u has an activation threshold θ_u and a node v becomes active only if $\Sigma_{u \in N^A_{inc}(v)} w(u, v) \geq \theta_u$ where $N^A_{inc}(v)$ and $w(u, v)$ are set of active incoming neighbors of v and edge weight of (u, v) respectively. For each edge $(u, v) \in E$, we assign edge weight $w(u, v)$ in LT model as follows.

$$w(u, v) = \begin{cases} \dfrac{1}{indegree(v)} & \text{Uniform} \\ (0, 1) & \text{Random} \\ \dfrac{c(u, v)}{\Sigma c(u, v)} & \text{Parallel} \end{cases} \tag{2}$$

where $c(u, v)$ is the number of parallel edges from u to v in multi-graph.

3.2.2. Independent cascade model

In this model, when a node u becomes active at time t_s, it has a only chance to activate its inactive neighbors v with activation probability p_{uv} at the time (t_s+1). If node v becomes active at the time $(t_s + 1)$ then it will never be inactive in future. The diffusion process terminates if no node is activated at time $(t_s + 1)$. For each edge $(u, v) \in E$, we assign edge weight $w(u, v)$ in IC model as follows.

$$w(u, v) = \begin{cases} [0.01, 0.1] & \text{Constant} \\ \dfrac{1}{indegree(v)} & \text{Weighted cascade} \\ \{0.001, 0.01, 0.1\} & \text{Tri-valency model} \end{cases} \tag{3}$$

3.3. Discrete particle swarm optimization

Kennedy and Eberhart [22] were first to introduced a population-based optimization meta-heuristic, viz. *particle swarm optimization* (PSO). PSO is inspired by the behaviors of fish schools and bird flocks. In PSO, a population (swarm) *pop* of individuals (particles), moves inside a bounded d-dimensional search space and cooperates with each-other to find best possible solution for a given problem. Each particle $j \in \{1, 2, 3, \dots, pop\}$ has two vectors, position $pos(j)$ and velocity $V(j)$. Let $pos(j) = (p_{j1}, p_{j2}, p_{j3}, \dots, p_{jd})$ and $V(j) = (v_{j1}, v_{j2}, v_{j3}, \dots, v_{jd})$ represents position and velocity vector of particle j respectively.

Originally, PSO was introduced to solve global continuous optimization problems. Later, Kennedy and Eberhart [42] proposed PSO for discrete optimization problems. Recently, a lot of work is done in PSO and many versions of PSO have been introduced. The version proposed by Shi and Eberhart [43], is used in most of the applications. The update rules for position and velocity are defined by Shi and Eberhart [44], as follows.

$$V(j) \leftarrow \delta V(j) + c_1 r_1 (pbest(j) - pos(j)) + c_2 r_2 (gbest - pos(j)) \tag{4}$$

$$pos(j) \leftarrow pos(j) + V(j) \tag{5}$$

where δ represents inertia weight, c_1 and c_2 are learn factors, $r_1 \in [0, 1]$ and $r_2 \in [0, 1]$ are random numbers, $pbest(j)$ is personal best position of particle j, and $gbest$ represents global best position in population.

3.4. Learning automata

Learning Automata (LA) is a decision-making machine with learning capability [45]. LA performs a finite set of actions α. Each action α_i is evaluated by a probability p_i and evaluation result generates a feedback β. The objective of LA is learn to find the optimal solution. LA is formulated as quadruple $\langle \alpha, \beta, p, T \rangle$, where $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_r\}$, $\beta = \{\beta_1, \beta_2, \beta_3, \dots, \beta_s\}$, $p = \{p_1, p_2, p_3, \dots, p_r\}$, and T represent an action set, response set, probability set, and the reinforcement process respectively. A linear reinforcement process [46] updates probability vector p for action $\alpha_i \in \alpha$ with $\beta \in \{0, 1\}$ as follows.

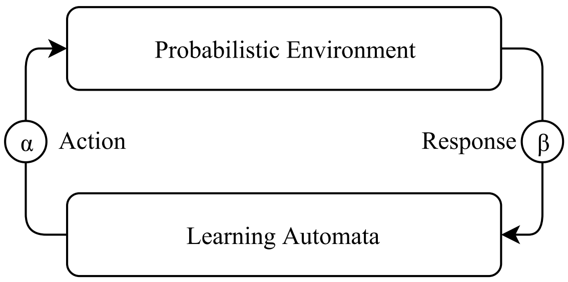


Fig. 1. The communication between learning automata and probabilistic environment.

$$p_i = \begin{cases} p_i + a_r[1 - p_i] & \beta = 0 \\ (1 - b_p)p_i & \beta = 1 \end{cases} \tag{6}$$

$$p_j = \begin{cases} (1 - a_r)p_j & \beta = 0 \\ \dfrac{b_p}{r-1} + (1 - b_p)p_j & \beta = 1 \end{cases} \tag{7}$$

where $a_r \in (0, 1)$ and $b_p \in (0, 1)$ represent reward and penalty parameters respectively. Fig. 1 shows the learning process of learning automata.

3.5. Problem definition

Given an influence graph $G = (V, E)$, an information diffusion model M, an integer k. Influence maximization process selects a seed set $S \subseteq V$ to maximize the influence spread in G, i.e.,

$$\sigma(S) = \underset{S^* \subseteq V \wedge |S^*| = k}{argmax} \; \sigma(S^*) \tag{8}$$

4. Proposed work

In this section, we present an algorithm named LAPSO-IM for maximizing influence spread in social networks. To maintain a trade-off between local and global search, LAPSO-IM uses LA to configure particles behavior in PSO. Firstly, we discuss the framework of the proposed work then we describe each component in detail.

Learning automata based PSO [23] has four phases: initialization, update, learning and final.

- **Initialization:** In this phase, an objective function is defined. Parameters values used in PSO and LA are assigned. The Initial velocity of each particle is assigned with zero. The position and personal best (pbest) of particles are assigned based on node degree centrality. Also, the global best (solution) is assigned using their calculated fitness values.
- **Update:** In this phase, iteration is performed based on the termination condition. All the particles in swarm move to a new position using their velocity. Each iteration is divided into sub-steps like, selection of LA action, velocity evaluation, fitness value estimation, finding *pbest* and *gbest* vectors.
- **Learning:** In this phase, the LA trains system based on their previous responses. It updates the probability vector based on learning machine response.

- **Final:** This phase performs constraints handling. To handle the boundary constraints, many methods exist.

Algorithm 1: LAPSO-IM(G,k):*Proposed Algorithm*

```
Input: Influence graph $G$, seed size $k$
Output: Seed set $S$
1  $j \leftarrow 1$                                        ▷ Initialization phase
2  Assign values to the parameters of PSO and LA
3  for $j \leq pop$ do
4      $pos(j) \leftarrow$ Select $k$ highest out-degree nodes
5      $pbest(j) \leftarrow$ Select $k$ highest out-degree nodes
6      $l \leftarrow 1$
7      for $l \leq k$ do
8          if $rand(0,1) < 0.5$ then
9              $p_{jl} \leftarrow replace(p_{jl}, V \setminus pos(j))$
10             $pbest_{jl} \leftarrow replace(pbest_{jl}, V \setminus pbest(j))$
11         $V_{jl} \leftarrow 0$
12         $l \leftarrow l+1$
13     Estimate particle fitness value $f(pos(j))$
14     $j \leftarrow j+1$
15 $gbest \leftarrow$ Select best fitness value particle $pos(i)$
16 $I \leftarrow 1$                                        ▷ Update phase
17 $p \leftarrow \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$
18 while $I \leq I_{max}$ do
19     $j \leftarrow 1$
20     for $j \leq pop$ do
21         LA select an action $\alpha_i$ using probability vector $p$
22         if $\alpha_i = local\ search$ then
23             Update velocity vector $V(j)$ using Eq. (13)
24         if $\alpha_i = global\ search$ then
25             Update velocity vector $V(j)$ using Eq. (14)
26         if $\alpha_i = global\ and\ local\ combined\ search$ then
27             Update velocity vector $V(j)$ using Eq. (15)
28         $pos_{prev}(j) \leftarrow pos(j)$
29         Update position vector $pos(j)$ using Eq. (17)
30         Update particle's fitness $f(pos(j))$
31         $pbest(j) \leftarrow$ LocalSearch($pbest(j)$, $pos(j)$)          ▷ See
           Algorithm 2
32         $\beta \leftarrow$ Generate LA response $\beta$ using Eq. (12)         ▷
           Learning
33         Update probability vector $p$ using Eqs. (6) & (7)
34         $j \leftarrow j+1$
35     $gbest' \leftarrow$ Select best fitness value particle $f(pos(i))$
36     $gbest \leftarrow max(gbest, gbest')$
37     $I \leftarrow I+1$
38 $S \leftarrow gbest$
39 Return $S$
```

Algorithm 2: LocalSearch*(pX,X)*

```
Input: Personal best vector $pX$, position vector $X$
Output: $Y$
1  $Y \leftarrow pX$
2  $Temp \leftarrow \{pX \setminus (pX \cap X)\}$
3  foreach $Y_j \in Temp$ do
4      $Y_j \leftarrow replace(Y_j, N(Y_j))$
5      if $f(f(Y) > f(pX)$ then
6          $pX \leftarrow Y$
7      else
8          $Y \leftarrow pX$
9  $Y \leftarrow pX$
10 Return $Y$
```

The principal components of the proposed algorithm LAPSO-IM are outlined as follow.

- *Swarm.* The swarm refers to the population *pop* of individuals (particles).
- *Particle.* The particles are agents of LAPSO-IM to perform optimization. The particles move towards the current best solution based on LA action α_i.
- *Fitness function.* In order to get the accurate influence spread of a seed set, the existing algorithms always need to run at least ten thousand time which is very time-consuming. As a result, Jiang et al. [24] proposed *expected diffusion value* (EDV) under IC model. In literature, there are several other objective functions of IM are presented such as MaxDegree [8], Degree Discount [15], Diffusion Degree [27], CUCB [47], and DILinUCB [47], etc. The objective functions MaxDegree, Degree Discount, and Diffusion Degree only consider node degree, and avoid their influence probabilities to their direct and indirect neighbors unlike EDV. Therefore, these objective functions generate poor quality seed than EDV with almost the same efficiency. CUCB and DILinUCB are based on pairwise influence feedback model, and generate seed nodes with approximate influence spread with low efficiency than EDV. Hence, it is essential to design a function that acquires a good trade-off between influence spread and running time to compute the influence spread of a node set. Therefore, we select EDV as fitness function of LAPSO-IM.
  EDV considers influence in the one-hope area (direct neighbors). The expected diffusion value of seed set S in the one-hope area is computed as follows.

$$EDV(S) = \sigma_0(S) + \sigma_1(S) = \quad k + \sum_{u \in N(S)\setminus S} (1 - \prod_{(u,v)\in E, v \in S} (1 - p_{uv})) \tag{9}$$

  where p_{uv} represents influence probability of u to v. In social networks, the influence spread of a user opinion is limited to within its local region like three-hope area [48], two-hope area [49]. Following these studies, we present an approximate influence estimation function $EDV^{(2)}(S)$ within the two-hope area. We can estimate the expected diffusion value in two-hope area $EDV^{(2)}(S)$, as follows.

$$EDV^{(2)}(S) = \sum_{i=0}^{i=2} \sigma_i(S) \tag{10}$$

$$EDV^{(2)}(S) = k + \sum_{u \in S, v \in N_{out}^{NA}(u)} p_{u,v} + \sum_{v \in \{N_{out}^{A}(S)\setminus S\}, w \in N_{out}^{NA}(v)} p_A(v).p_{v,w} \tag{11}$$

  where $N_{out}^{NA}(u)$ represents in-active out-neighbors of u, and $N_{out}^{A}(S)$ represents active out-neighbors of S. The fitness function of LAPSO-IM is given by local influence spread function $EDV^{(2)}(S)$ depicted in Eq. (11).
- *Learning automata.* Each LA performs one of three actions: *global search*, *local search*, and *global and local combined search* [23]. Based on action α_i, the reinforcement signal β is generated. The value of response/feedback by the probabilistic environment is calculated by Eq. (12).

$$\beta = \begin{cases} 0 & f(pos(j)) > f(pos_{prev}(j)) \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

- *Velocity.* The velocity of a particle j is represented by vector $V(j) = (v_{j1}, v_{j2}, v_{j3}, \ldots, v_{jk})$ where $k = |S|$. Suppose that LA selects action *local search* (LS) then particle moves toward the best position with $\delta = 0$ and velocity $V(j)$ of particle j is calculated as follows.

$$V(j) \leftarrow F(r_1(a + \frac{1}{R_{it}})(pbest(j) \cap pos(j)) + (b - \frac{1}{R_{it}})(pbest(j) \cap pos(j))) \tag{13}$$

where $a, b \in [0, 1]$ are the weight index (learn factor), $r_1 \in [0.6, 1.2]$ represents random number, and $R_{it} = (I_{max} + 1) - I$. I_{max} and I denote maximum number of iteration and current iteration number respectively. If LA selects action *global search* (GS) then velocity $V(j)$ is calculated as follows.

$$V(j) \leftarrow F(r_1(a + \frac{1}{R_{it}})(pbest(j) \cap pos(j)) + cr_2(gbest \cap pos(j))) \tag{14}$$

where c denotes acceleration constant, and $r_2 \in [0, 1]$ represents random number. If LA selects action *global and local search combined* (GLC) then velocity $V(j)$ is calculated as follows.

$$V(j) \leftarrow F(\delta V(j) + r_1(a + \frac{1}{R_{it}})(pbest(j) \cap pos(j)) + (b - \frac{1}{R_{it}})(pbest(j) \cap pos(j)) + cr_2(gbest \cap pos(j))) \tag{15}$$

where $\delta \in [0, 1]$ is inertia weight. Parameter δ is used to control degree of local and global search. The operator $\cap$ used in same sense as intersection operator. For example, suppose two vectors X and Y are {A,D,F} and {A,B,D} respectively, then $X \cap Y = \{0, 1, 0\}$. The argument of function $F(.)$ is a position vector. Suppose that argument of $F(.)$ is $pos(i) \in \{pos(1), pos(2), \dots, pos(pop)\}$ then function $F(pos(i))$ is denoted as $\{f_1(p_{i1}), f_2(p_{i2}), \dots, f_k(p_{ik})\}$. The value of $f_l(p_{il}) \in F(pos(i))$ is computed as follows.

$$f_l(p_{il}) = \begin{cases} 0 & p_{il} < 1 \\ 1 & \text{otherwise} \end{cases} \tag{16}$$

For example, the argument of $F(.)$ is (1.2,0.78,2.3) then $F(1.2, 0.78, 2.3) = (1, 0, 1)$.

- *Position.* The position of a particle j is represented by vector $pos(j) = (p_{j1}, p_{j2}, p_{j3}, \dots, p_{jk})$ where $k = |S|$. Each particle's position give a solution. In each iteration, position of particle j is estimated as follows.

$$pos(j) = pos(j) \otimes V(j) \tag{17}$$

where the operator $\otimes$ is defined as $pos(j) \otimes V(j) = pos'(j) = \{p'_{j1}, p'_{j2}, \dots, p'_{jk}\}$ and $p'_{jl} \in pos'(j)$ is computed as follows.

$$p'_{jl} = \begin{cases} p_{jl} & v_{jl} = 0 \\ replace(p_{jl}, N) & v_{jl} = 1 \end{cases} \tag{18}$$

where $N \in G$. For example, $pos(j)$ and $V(j)$ are {B,F,G} and {0,1,0} respectively, then $pos(j) = (B, D, G)$.

- *pbest(j).* The personal best position of particle j is defined as $pbest(j) = \{pbest_{j1}, \dots, pbest_{jk}\}$.
- *gbest.* The global best position of swarm is defined as $gbest = \{gbest_1, \dots, gbest_k\}$ which gives best possible solution of IM problem.

5. Algorithm

The main Algorithm 1 takes two inputs, an influence graph G and the size of seed set k. LAPSO-IM performs the local and global search based on LA response in influence graph G using PSO and finds k most influential users. Fig. 2 presents the flow chart for working of the algorithm. First of all, line 1 assigns j to 1. Line 2 assigns values of required parameters of PSO and LA likes population size pop, the maximum number of iteration I_{max}, inertia weight δ, weight indexes (a, b), reward parameter a_r, and penalty parameter b_p etc. The **for** loop in lines 3–14, initializes position, velocity, and $pbest$ vectors of particle j. Lines 4–5 assign position pos and personal best $pbest$ vector with k highest out-degree nodes. Line 6 initializes l with 1. The **for** loop in lines 7–12, promotes diversity in positions of particles. Line 13 computes fitness of particle j. Line 14 performs increment in l. Line 15 finds solution vector $gbest$. Line 16 initializes iteration number I with 1. Line 17 initializes the probability vector p of learning automata. The **while** loop in lines 18–37, finds solution $gbest$ iteratively. Line 19 initializes j with 1. The **for** loop in lines 20–34, updates position and velocity vector of particles repeatedly. Line 21 selects an action $\alpha_i \in \alpha$ based on probability vector p. Lines 22–23 check, if selected action is *local search* then velocity is estimated using Eq. (13). Lines 24–25 check, if selected action is *global search* then velocity is estimated using Eq. (14). Lines 26–27 check, if selected action is *global and local combined search* then velocity is estimated using Eq. (15). Line 28 stores previous position vector $pos(j)$ of particle j before updating the position vector. Line 29 updates position vector $pos(j)$ of particle j. Lines 30 calculates particle fitness. Line 31 updates local best position vector $pbest(j)$ of particle j. Line 32 calculates the reinforcement (response) signal β using Eq. (12). Line 33 updates probability vector p of learning automata. Line 34 performs the increment operation on j. Line 35 estimates $gbest$ of the current iteration. Line 36 finds global best position vector $gbest$ until current iteration. Line 37 performs the increment operation on I. LAPSO-IM repeats the iterations of the **while** until the termination condition $I \leq I_{max}$ is satisfied. Line 38 assigns $gbest$ to seed set S. Lastly, line 39 returns the seed set as the solution of given problem.

Algorithm 2 takes two position vectors pX and X as input. It performs local search and finds best position vector $pbest(j)$ for each particle j. First of all, line 1 stores position vector pX into Y. Line 2 selects nodes from vector pX which are not present in vector X. The **foreach** loop in lines 3–8, finds personal/local best position vector. Line 4 replaces Y_j with its neighbors. Lines 4–8 update local position vector. Line 9 assigns vector pX to Y. Lastly, Line 10 returns local best position vector Y.

5.1. Applying the algorithm

To explain the working of the proposed algorithm, we take an example graph as shown in Fig. 3. Here, we consider LAPSO-IM system as a complete undirected graph $G(V, E)$ where V represents particles in swarm and E represents edges between them. Each particle $P(j)$ connects with other particles i.e, $|E| = \frac{|V|(|V|-1)}{2}$. Suppose that population of swarm $pop = 5$.

Initialization: Suppose, $pop \leftarrow 5$, $I_{max} \leftarrow 3$, $\delta \leftarrow 0.6$, $a \leftarrow 0.8$, $b \leftarrow 0.6$, $c \leftarrow 0.5$, $a_r \leftarrow 0.8$, $b_p \leftarrow 0.6$, $k \leftarrow 2$, $\beta \leftarrow \{0, 1\}$, and diffusion model $M \leftarrow WC$. Initially, velocity $V(j)$ of each particle P_j is assigned to zero. Position $pos(j)$ and personal best $pbest(j)$ are initialized by k highest degree nodes. To promote swarm diversity, LAPSO-IM uses replace method (lines 7–12). Let us assume that particle P_1 initializes $pos(1)$ and $pbest(1)$ with (B, E) and (D, E) respectively. Fitness of particle P_1 and P_2 are calculated as $f(1) = 2 + (0.3 \oplus 0.3) + (0 + 0) = 2.3$ and $f(2) = 2 + ((0.5 + 0.4 + 0.3) + 0.3) + (0 + 0 + 0 + 0.3 * 0.1) = 3.53$ respectively. Similarly, we estimate fitness of particle P_3, P_4, and P_5 as shown in Table 2. Probability vector is initialize with $p(LS, GS, CLC) = (1/r, 1/r, 1/r) = (0.33, 0.33, 0.33)$.

Iteration: Iteration phase starts at $I = 1$. Each particle P_i updates position and velocity vectors as follows.

Table 2
LAPSO-IM update phase in running example.

Input: *pop* $\leftarrow 5$, $I_{max} \leftarrow 3$, $\delta \leftarrow 0.6$, $a \leftarrow 0.8$, $b \leftarrow 0.6$, $c \leftarrow 0.5$, $a_r \leftarrow 0.8$, $b_p \leftarrow 0.6$, $k \leftarrow 2$

I	j	$V(j)$	$pos(j)$	$f(j)$	$pbest(j)$	$p(LS, GS, GLC)$	β	α_i	r_1	r_2	$gbest$
0	1	(0,0)	(B,E)	2.30	(D,E)	(0.33, 0.33, 0.33)		–	–	–	(F,I)
	2	(0,0)	(D,C)	3.53	(E,G)	(0.33,0.33,0.33)	–	–	–	–	
	3	(0,0)	(D,H)	3.77	(D,B)	(0.33,0.33,0.33)	–	–	–	–	
	4	(0,0)	(F,I)	3.91	(F,J)	(0.33,0.33,0.33)	–	–	–	–	
	5	(0,0)	(D,A)	3.20	(H,F)	(0.33,0.33,0.33)	–	–	–	–	
1	1	(0,0)	(B,E)	2.30	(D,E)	(0.13,0.43,0.43)	1	LS	0.4	0.8	(D,J)
	2	(1,1)	(D,J)	4.50	(E,H)	(0.02,0.08,0.80)	0	GLC	0.7	0.2	
	3	(0,0)	(D,H)	3.77	(D,G)	(0.30,0.33,0.32)	1	GLC	0.6	0.5	
	4	(0,0)	(F,I)	3.91	(F,J)	(0.42,0.13,0.43)	1	GS	0.4	0.6	
	5	(0,0)	(D,A)	3.20	(D,J)	(0.46,0.35,0.17)	1	GLC	0.3	0.4	
2	1	(1,0)	(G,E)	2.43	(D,E)	(0.89,0.07,0.03)	0	LS	0.8	0.6	(D,J)
	2	(0,0)	(D,J)	4.50	(D,H)	(0.35,0.32,0.31)	1	LS	0.6	0.4	
	3	(0,1)	(D,I)	3.46	(D,H)	(0.14,0.43,0.42)	1	LS	0.7	0.4	
	4	(0,1)	(F,C)	3.95	(F,J)	(0.03,0.80,0.08)	0	GS	0.8	0.5	
	5	(0,1)	(D,G)	3.33	(D,J)	(0.01,0.96,0.02)	0	GS	0.6	0.5	
3	1	(0,0)	(G,E)	2.43	(D,E)	(0.30,0.38,0.31)	1	GS	0.4	0.8	(D,J)
	2	(0,1)	(D,H)	3.77	(D,J)	(0.42,0.15,0.43)	1	GS	0.9	0.6	
	3	(0,1)	(DJ)	4.50	(DH)	(0.08,0.03,0.80)	0	GLC	0.7	0.5	
	4	(0,1)	(F,A)	3.47	(F,J)	(0.31,0.30,0.32)	1	GLC	0.3	0.6	
	5	(0,1)	(D,H)	3.77	(D,J)	(0.06,0.06,0.86)	0	GLC	0.2	0.7	

Output: Seed set $\leftarrow \{D, J\}$, Active set $\leftarrow \{A, B, D, E, G, H, I, J\}$, Influence spread $\leftarrow 4.50$

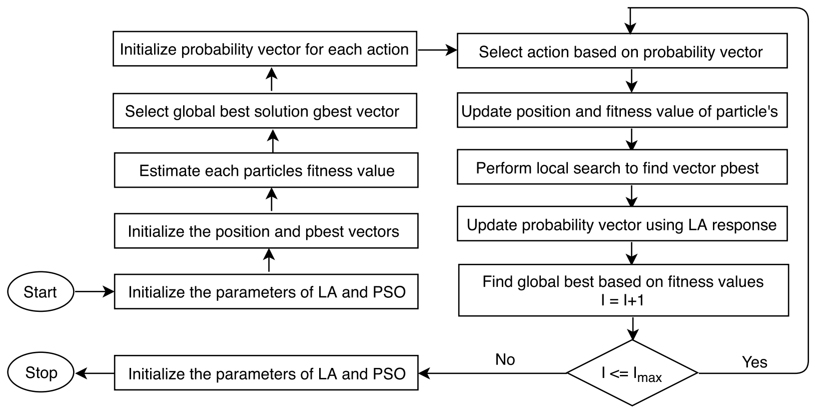

Initialize probability vector for each action
Select action based on probability vector
Select global best solution gbest vector
Update position and fitness value of particle's
Estimate each particles fitness value
Perform local search to find vector pbest
Initialize the position and pbest vectors
Update probability vector using LA response
Start
Initialize the parameters of LA and PSO
Find global best based on fitness values
I = I+1
Stop
Initialize the parameters of LA and PSO
No
I <= I_max
Yes

Fig. 2. The working of our proposed algorithm.

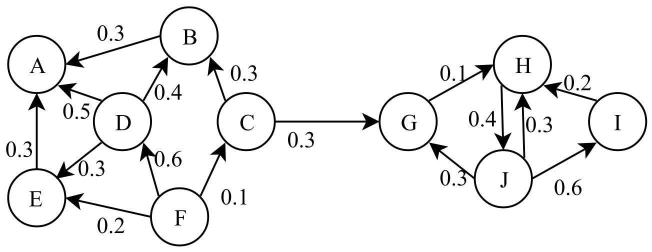


Fig. 3. A running example graph.

- *Particle* P_1. Suppose LA chooses an action $\alpha_i = LS$ based on probability vector p then velocity $V(1)$ is calculated using Eq. (13). So $V(1)$ is estimated as $V(1) = F(r_1(a + \frac{1}{R_{it}})(pbest(1) \cap pos(1)) + (b - \frac{1}{R_{it}})(pbest(1) \cap pos(1))) = F(0.4(0.8 + \frac{1}{3+1-1})(DE \cap BE) + (0.6 - \frac{1}{3+1-1})(DE \cap BE)) = F(0.452(1, 0) + 0.27(1, 0)) = (0, 0)$. Using Eq. (17), position of P_1 is estimated as $pos(1) = pos(1) \otimes V(1) = (B, E) \otimes (0, 0) = (B, E)$. $f(1) = 2.3$, $pbest = (D, E)$. Now we estimate response β using Eq. (12), $f(1) \leq f_{prev}(1)$, $\beta = 1$. Probability vector p is updated using Eqs. (6)–(7). $p = ((1-b_p)p_1, \frac{b_p}{r-1} + (1-b_p)p_1, \frac{b_p}{r-1} + (1-b_p)p_1) = ((1-0.6)*0.33, 0.3 + (1-0.6)*0.33, 0.3 + (1-0.6)*0.33) = (0.13, 0.43, 0.43)$.
- *Particle* P_2. LA chooses action $\alpha_i = GLC$ using vector p. Using Eq. (15), velocity is computed as $V(2) = F(0.6(0, 0) + 0.7(0.8 + 0.33)(EG \cap DC) + (0.6 - 0.33)(EG \cap DC) + 0.5 * 0.2(DH \cap DC)) = 0.79(1, 1) + 0.27(1, 1) + 0.1(0, 1) = (1, 1)$. Position of P_2 is $pos(2) = (D, C) \otimes (1, 1) = (D, J)$. Fitness value of P_2 is estimated as $f(2) = 2 + (0.5 + 0.4 + 0.3) + (0.6+0.4+0.3) = 4.5$. $pbest = (E, H)$ and $\beta = 0$. Probability vector is calculated as $p = ((1 - 0.8) * 0.13, (1 - 0.8) * 0.43, 0.43 + 0.8 * (1 - 0.43)) = (0.02, 0.08, 0.80)$.
- *Particle* P_3. LA selects action $\alpha_i = GLC$ using vector p. Using Eq. (15), velocity is computed as $V(3) = F(0.6(0, 0) + 0.6(0.8 + 0.33)(DB \cap DH) + (0.6 - 0.33)(DB \cap DH) + 0.5 * 0.5(DH \cap DH)) = 0.67(0, 1) + 0.27(0, 1) + 0.25(0, 0) = (0, 0)$. Position is $pos(3) = (D, H) \otimes (0, 0) = (D, H)$. $f(2) = 3.77$, $pbset(3) = (D, G)$, and $\beta = 1$. p is calculated as $(0.3 + (1 - 0.6) * 0.02, 0.3 + 0.4 * 0.08, (1 - 0.6) * 0.8) = (0.30, 0.33, 0.32)$.
- *Particle* P_4. LA chooses action $\alpha_i = GS$ using vector p. Using Eq. (14), velocity is computed as $V(4) = F(0.4(0.8 + 0.33)(FJ \cap FI) + 0.5 * 0.6(DH \cap FI)) = 0.45(0, 1) + 0.3(0, 0) = (0, 0)$. Position is $pos(4) = (F, I) \otimes (0, 0) = (F, I)$. $f(4) = 3.91$, $pbset(4) = (F, J)$, and $\beta = 1$. p is calculated as

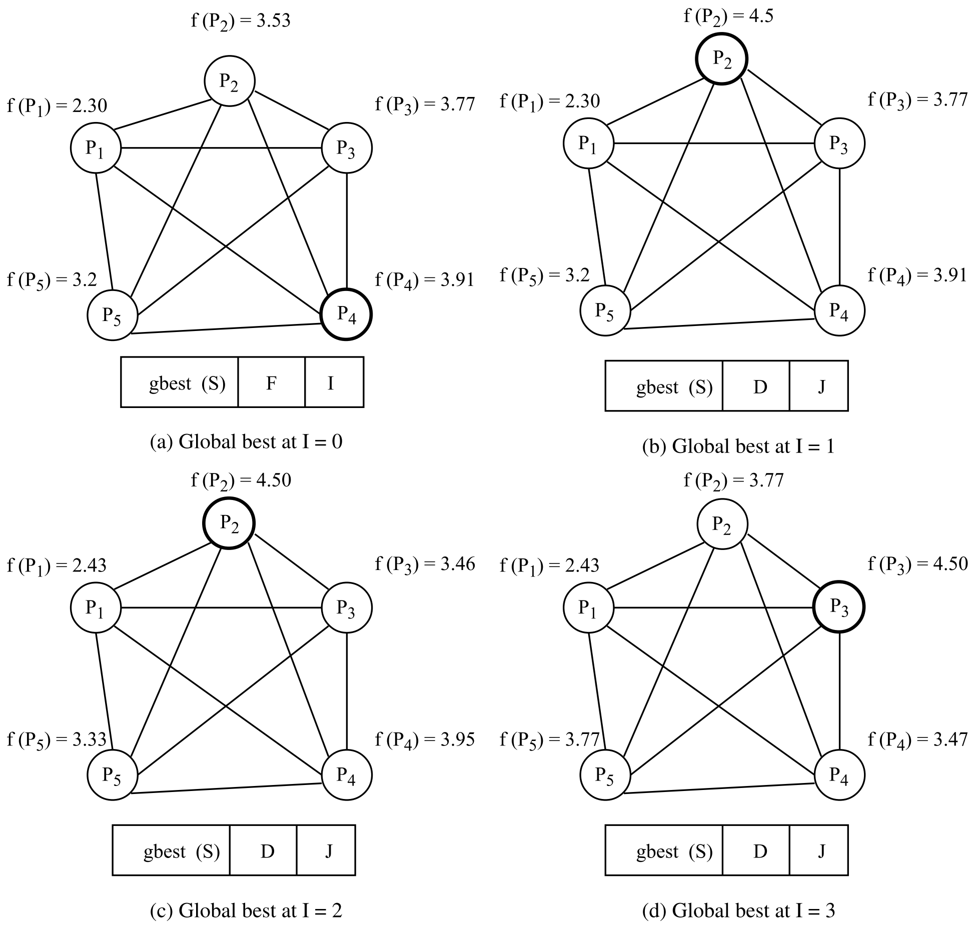


Fig. 4. Working example of the LAPSO-IM when pop = 5.

$(0.3 + (1 - 0.6) * 0.3, (1 - 0.6) * 0.33, 0.3 + 0.4 * 0.32) = (0.42, 0.13, 0.43)$.

- *Particle* P_5. LA selects action $\alpha_i = GLC$ based on vector p. Using Eq. (15), velocity is computed as $V(5) = F(0.6(0, 0) + 0.3(0.8 + 0.33)(HF \cap DA) + (0.6 - 0.33)(HF \cap DA) + 0.5 * 0.4(DH \cap DA)) = 0.33(1, 1) + 0.27(1, 1) + 0.2(0, 1) = (0, 0)$. Position is $pos(5) = (D, A) \otimes (0, 0) = (D, A)$. $f(5) = 3.20$, $pbset(5) = (D, J)$, and $\beta = 1$. p is calculated as $(0.3 + (1 - 0.6) * 0.42, 0.3 + 0.4 * 0.013, (1 - 0.6) * 0.43) = (0.46, 0.35, 0.172)$.

The process continues until $I \leq I_{max}$ in the same manner. Table 2 gives the detailed description of each iteration.

Final: Finally the algorithm returns the desired output $gbest = \{D, J\}$ as seed set.

Fig. 4 shows working of LAPSO-IM system. Fig. 4a shows the fitness of each particle at iteration $I = 0$. Particle P_4 has highest fitness value so position of P_4 is considered as current global best *gbest*. Fig. 4b shows that particle P_2 has highest fitness value at iteration $I = 1$, so that $gbest = \{D, J\}$. Similarly, Figs. 4c and 4d show the *gbest* at $I = 2$ and $I = 3$ respectively. The final global best *gbest* is considered as seed set (solution).

5.2. Complexity analysis

In this section, we analyze the time complexity of the proposed algorithm. For each particle, the initialization phase (lines 1–17) needs $O(k|V|)$, $O(k|V|)$, and $O(k.D_{avg}(N)^2)$ time to initializes position, *pbest*, and fitness respectively, where $D_{avg}(N)$ is the average degree of the network. To initialize the *gbest*, it takes $O(n)$ time, where $|pop| = n$. Hence, the time complexity of initialization phase is $O(n(k|V| + k|V| + k.D_{avg}(N)^2) + n)$. In update phase, velocity of each particle is estimated in $O(k. \log k)$ time (lines 21–27). The position of a particle is calculated in $O(k)$ time (line 29). The fitness of each particle is calculated in $O(kD_{avg}(N)^2)$ time (line 30). The personal best *pbest* is updated in time $O(k.D_{avg}(N))$ (line 31). The probability vector is updated in $O(r)$ time (line 33). The global best *gbest* is updated in $O(k)$ time (lines 36–37). The overall time complexity of update phase is $O(I_{max}(n(k. \log k + k + k.D_{avg}(N)^2 + k.D_{avg}(N) + r) + k))$. Therefore, the worst case time complexity of LAPSO-IM is $O(I_{max}.n.k.(\log k + D_{avg}(N)^2) + n.k.|V|)$.

5.3. Compared with the state-of-the-art methods

The simulation-based approaches [9–12,26] have the benefit of *model generality*, i.e., these approaches easily incorporate any traditional diffusion models. These approaches used time-consuming MC simulations for influence spread estimation. Therefore, these approaches have a major drawback in terms of *time-efficiency*. Hence, simulation based IM approaches are best suited for small-scale networks. The heuristic-based IM approaches [14–18,28,30] and the meta-heuristic approaches [32,33] have the benefit of *practical efficiency* and lack of *theoretical guarantee*. These approaches are unstable, i.e., a minor change in underlying network results a drastic change in seed set and corresponding influence spread. The sketch-based IM approaches [19–21] guarantee the *theoretical efficiency*. The performance of these approaches are bounded with its diffusion models i.e., these approaches are not adaptable to wider range of diffusion models. LAPSO-IM meta-heuristic also has the benefit

Table 3
Comparison of the characteristics of LAPSO-IM algorithm with the existing IM algorithms — 1.

Algorithm	Time complexity	Approximation	Problem solving perspective	State-of-the-art algorithms	Base algorithm
Greedy [8]	$O(kNMI)$	$1-1/e-\varepsilon$	Spread simulation	MaxDegree, Central & Random	–
Knapsack Greedy [25]	$O(N^5)$	$1-1/e-\varepsilon$	Spread simulation	–	Greedy
SP1M [14]	$O(kNM)$	$1-1/e$	Influence path	Degree, PageRank & Closeness	–
CELF [9]	$O(kNMI)$	$1-1/e-\varepsilon$	Sub-modularity	Greedy	Greedy
Degree Discount [15]	$O(k\log N+M)$	N.A.	Spread simulation	CELF, Greedy & Random	High Degree
NewGreedy [15]	$O(kIM)$	N.A.	Snapshots	CELF, Greedy & Random	High Degree
TW Greedy [10]	$O(kNMI)$	$1-1/e-\varepsilon$	Spread simulation	SCG, KKG & High Degree	Greedy
MIA/PMIA [16]	$O(Nt_{i\theta}+kn_{o\theta}n_{i\theta}(n_{i\theta}+\log N))$	$1-1/e$	Influence path	Greedy, Random, DD & PageRank	SP1M
LDAG [17]	$O(Nt_{\theta}+kn_{\theta}m_{\theta}(m_{\theta}+\log N))$	N.A.	Score estimation	Greedy, SPIN, DD & PageRank	–
CGA [11]	$O(M+IM_C(N(Z-C)+k(C+N_C)))$	$1-e^{-1/(1+\delta_C)}$	Community based	DD, MG & Random	–
CELF++ [12]	$O(kNMI)$	$1-1/e-\varepsilon$	Sub-modularity	CELF	CELF
Diffusion Degree [27]	$O(N+M)$	N.A.	Centrality based	DD & High Degree	High Degree
SIMPATH [30]	$O(klNP_{\theta})$	N.A.	Score estimation	High Degree, CELF & PageRank	LDAG
IRIE [31]	$O(k(n_{o\theta}k+M))$	N.A.	Score estimation	Greedy & PMIA	–
IPA [28]	$O(\frac{NO_v n_{vu}}{c}+k^2(\frac{O_v n_{vu}}{c}+(c-1)))$	N.A.	Influence path	Greedy, DD & Random	PMIA
StaticGreedy [19]	$O(\frac{kMN^2\log\binom{N}{k}}{\varepsilon^2})$	$1-1/e-\varepsilon$	Snapshots	CELF, SP1M, DD & High Degree	PMIA
PRUNEDMC [20]	$O(\frac{kMN^2\log\binom{N}{k}}{\varepsilon^2})$	$1-1/e-\varepsilon$	Snapshots	IRIE, Random, PMIA & Degree	Greedy
TIM [21]	$O(\frac{k(M+N)\log N}{\varepsilon^2})$	$1-1/e-\varepsilon$	Reverse reachability	CELF++, IRIE & SIMPATH	–
DPSO [33]	$O(k^2\log kn\bar{D}^2)$	N.A.	Swam optimization	Degree, CELF++ & SAEDV	PSO
LAIM [26]	–	$1-1/e-\varepsilon$	Learning based	Degree, CELF & Random	Greedy
LAPSO-IM	$O(I_{max}.n.k.(\log k+D^2)+n.k.N)$	–	Swam optimization	DD, CELF++, IMLA, & DPSO	DPSOLA

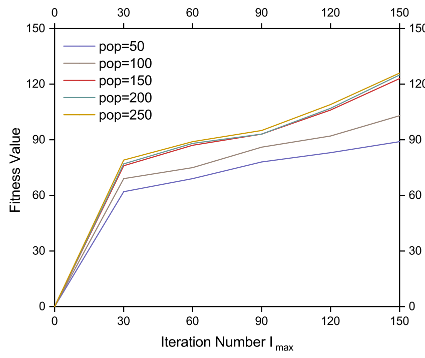
Table 4
Comparison of the characteristics of LAPSO-IM algorithm with the existing IM algorithms — 2.

Algorithm	Diffusion model				Category				Network	
	Linear threshold	Independent cascade	Triggering	Continuous time-aware	Simulation	Heuristic	Meta-heuristic	Sketch	Single	Static
Greedy [8]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Knapsack Greedy [25]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
SP1M [14]	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓
CELF [9]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Degree Discount [15]	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓
NewGreedy [15]	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
TW Greedy [10]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
MIA/PMIA [16]	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓
LDAG [17]	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓
CGA [11]	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓
CELF++ [12]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Diffusion Degree [27]	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓
SIMPATH [30]	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓
IRIE [31]	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓
IPA [28]	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓
StaticGreedy [19]	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
PRUNEDMC [20]	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
TIM [21]	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓
DPSO [33]	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓
LAIM [26]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
LAPSO-IM	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓

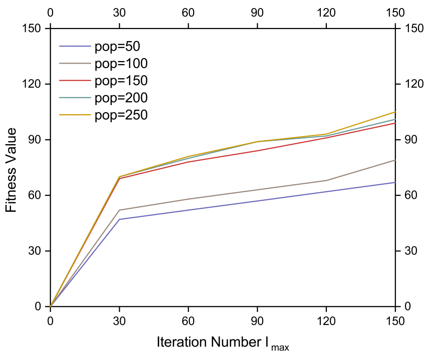
of *practical efficiency* and *model generality* while lacks *theoretical guarantee*. LAPSO-IM produces better influence spread than existing heuristics and meta-heuristics with comparable efficiency. The proposed algorithm generates seed set efficiently with approximate influence spread than simulation and sketch-based approaches.

Table 3 provides a theoretical analysis of the above-discussed IM approaches. Table 3 gives the primary information such as the name of the algorithm, approximation ratio, and problem-solving perspective, etc. The column *name of the algorithm* states the name of the algorithm with its reference. The columns *Time Complexity*, *Approximation*, and *Problem Solving Perspective* give the worst-case time complexity of the algorithm, the approximation ratios of the algorithm (N.A. indicates no approximation guaranteed), and solution framework perspective respectively. The columns *State-of-the-art Algorithm* and *Base Algorithm* give the name of the state-of-the-art methods and the name of the base algorithm for the corresponding algorithm.

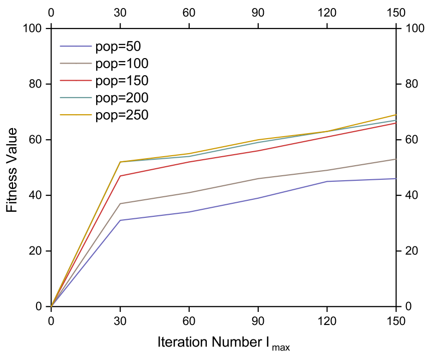
Table 4 compares the characteristics of LAPSO-IM algorithm with the existing IM algorithms. Table 4 gives the information of their diffusion model, category, and type of network. The column *name of the algorithm* states the name of the algorithm with its reference. The columns *Diffusion Model* show that whether the corresponding algorithm works under classic diffusion models (✓= work and ✗= not work). The columns *Category* explains the category where the corresponding algorithm belongs based on problem-solving strategy of the algorithm (✓= belongs and ✗= not belongs). The columns *Network* explains the type of network where the algorithm can be applied (✓= applied and ✗= not applied).



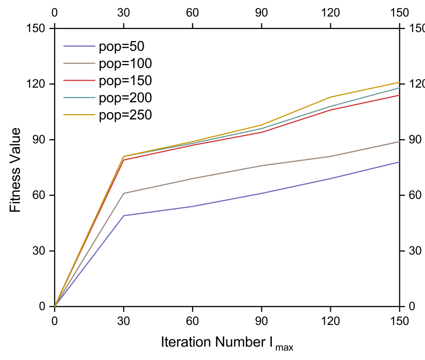
(a) NetInfective



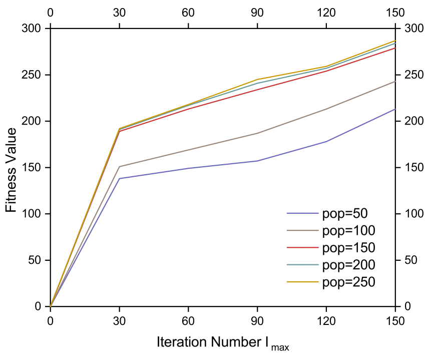
(b) NetScience



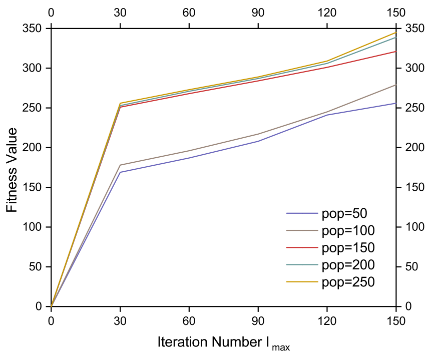
(c) Ca-GrQc



(d) NetHEPT



(e) Gnutella30



(f) NetPHY

Fig. 5. Variation of fitness value with distinct swarm population *pop*.

6. Evaluation

6.1. Dataset

NetInfective[1] [50] network describes people's behavior during face to face interaction at INFECTIOUS exhibition. Gnutella30[2] [51] is a peer-to-peer file sharing network. The other networks NetScience[1] [52], Ca-GrQc[3] [53], NetHEPT[1] [15], and NetPHY[1] [54] are co-author networks and describe authors relationship among various research domain. Table 5 describes the statistical information of the real-world datasets.

Table 5
Statistical information of the real-world datasets.

Dataset	$\|V\|$	$\|E\|$	$D_{avg}(N)$	Type
NetInfective [50]	410	17 298	84.38	Dense
NetScience [52]	1 589	2 742	3.45	Sparse
Ca-GrQc [53]	5 242	14 496	5.53	Sparse
NetHEPT [15]	15 233	31 398	4.12	Sparse
Gnutella30 [51]	36 682	88 328	4.81	Sparse
NetPHY [54]	37 154	174 161	9.38	Sparse

6.2. Algorithm to compare

- Degree Discount: This approach is based on the concept that after a node is selected as seed node then it will no longer be influenced by its neighbors [15].
- Diffusion Degree: This heuristic method combined the user's influence to its direct neighbors and from neighbors to rest of the network [27].
- CELF++: It is an extension of hill-climbing greedy approach. This algorithm selects k users using the sub-modular property of objective function [12].
- IMLA: This is the learning automata based IM approach [26]. IMLA is more time efficient than CELF++ with almost same influence spread because it reduces the number of time-consuming MC simulations. However, time efficiency is still an issue.
- DPSO: This is a discrete particle swarm optimization based approach [33]. It uses only local search. Therefore, it reduces influence spread in the network.
- LAPSO-IM: This is our proposed algorithm.

6.3. Parameter setting

We consider the size of seed set $k = 50$ to set the various parameters value of LAPSO-IM under IC diffusion model.

6.3.1. The size of the population and the number of iterations

In order to analyze the performance of LAPSO-IM with distinct population size *pop* and the number of iteration I_{max}, we set learning factors a and b to 1.2, and inertia weight w to 0.8. Figs. 5 and 6 show the fitness value of LAPSO-IM with distinct values of I_{max} and *pop* respectively.

Fig. 5 shows that the fitness values are increased with swarm population *pop* within 150 iterations. But LAPSO-IM generates very approximate fitness value with $pop > 150$ for six real-world social networks. To achieve a trade-off between running time and performance, we set swarm size $pop = 150$.

Fig. 6 shows the fitness value curves in six real-world networks within 600 iterations. Fitness curves of Fig. 6 illustrate

[1] https://arxiv.org/.
[2] https://snap.stanford.edu/data/p2p-Gnutella30.html.
[3] http://snap.stanford.edu/data/ca-GrQc.html.

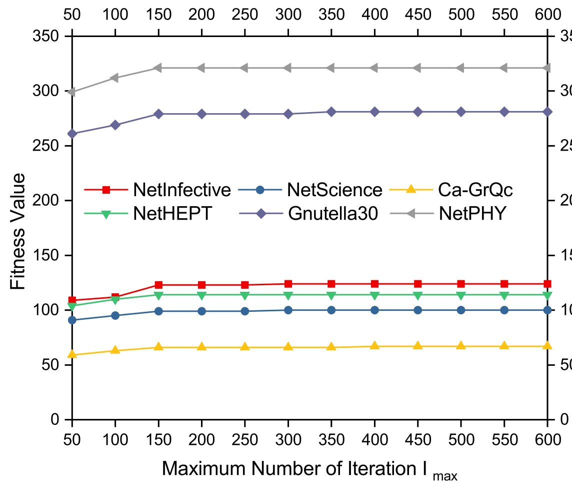


Fig. 6. Iteration number I_{max}.

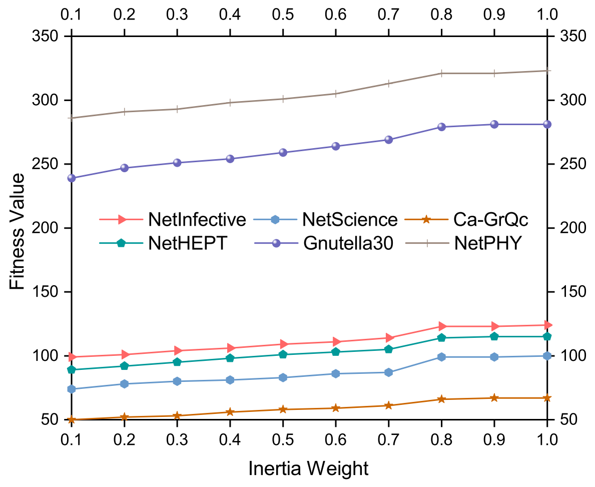


Fig. 7. Inertia weight δ.

that LADPSO-IM has converged within 150 iterations for all social networks. In order to achieve a trade-off between running time and performance, $I_{max} = 150$ is a reasonable choice.

6.3.2. *The inertia weight and learn factors*

In order to verify the values of the inertia weight and learn factors, *pop* and I_{max} are both set to 150. Fig. 7 shows that the fitness values are raised with inertia weight δ for six real-world social networks. But the fitness values with $\delta \geq 0.8$ are very approximate. Thus, we set $\delta = 0.8$ for performance analysis on social networks.

As for the learn factors, we vary the values of learn factors and fix the value of $\delta = 0.8$. As shown in Fig. 8, the fitness values are raised with the increase in learn factors a and b in all six social networks. Therefore, both a and b are set to 1.2.

6.3.3. *The reward and penalty parameter*

In order to set the values of the reward a_r and penalty b_p parameters, we keep the swarm size *pop* = 150, number of iterations I_{max} = 150, inertia weight δ = 0.8, learn factors $a = b = 1.2$, and vary the values of reward and penalty parameter between [0.1,0.9]. As shown in Fig. 9, the fitness values are increased with the increase of a_r and b_p. But the fitness values with $a_r \geq 0.6$ and $b_p \geq 0.6$ are very approximate. Thus, we set both a_r and b_p to 0.6.

6.4. *Experimental results*

6.4.1. *Quality*

Quality in IM problem equates to the influence spread of seed nodes S in the social network. An algorithm with higher influence spread is known as higher quality algorithm. We select 5 distinct set of seed users of size 10, 20, 30, 40, and 50 for six real-world social networks NetInfective, NetScience, Ca-GrQc, NetHEPT, Gnutella30, and NetPHY datasets for quality analysis of LAPSO-IM.

Figs. 10, 11, and 12 show the influence spread results of LAPSO-IM with the best suited state-of-the-art algorithms under IC, WC, and LT diffusion models respectively. As shown in Figs. 10, 11, and 12, LAPSO-IM outperforms both heuristic approaches Degree Discount and Diffusion Degree in terms of influence spread under each diffusion models for all networks. This is because both of heuristic approaches do not consider any feature, context or information except their node degree. LAPSO-IM also performs better than base method DPSO under each diffusion models for

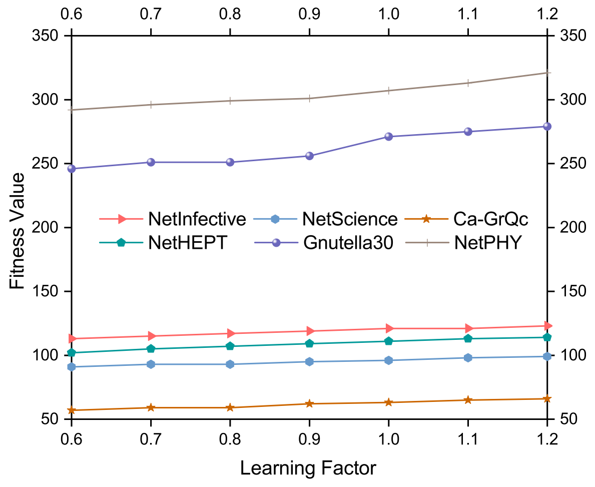


Fig. 8. Learning factors a and b.

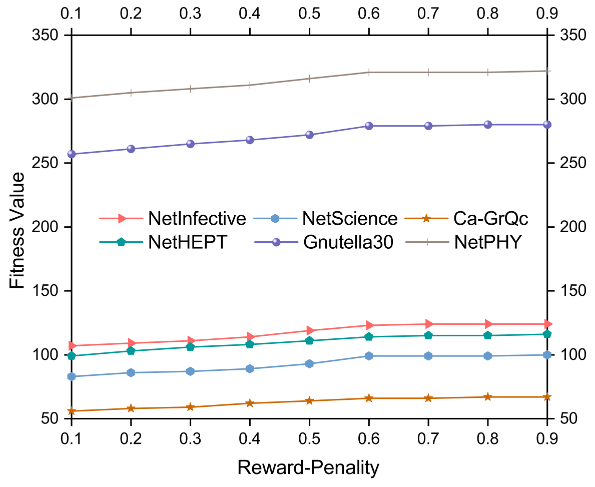
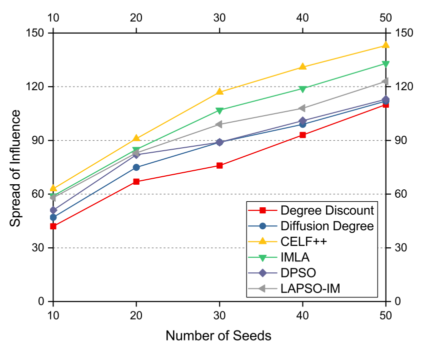
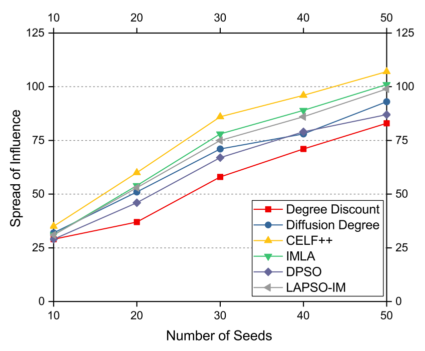


Fig. 9. Variation of fitness value with distinct linear reward-penalty ($a_r = b_p$) values.

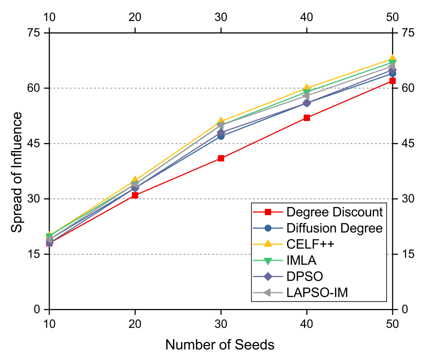
all networks. The reason behind this is that DPSO uses only local search strategy, while LAPSO-IM performs local search, global search as well as the combination of both.



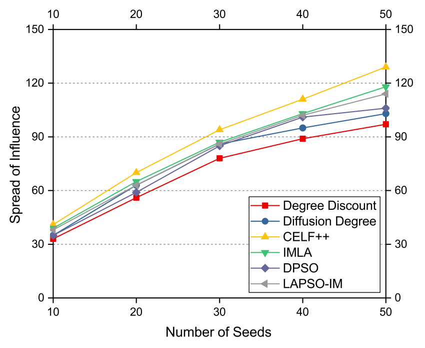
(a) NetInfective



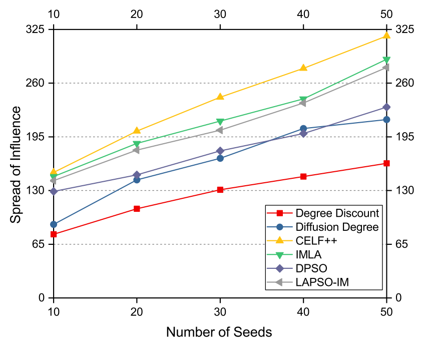
(b) NetScience



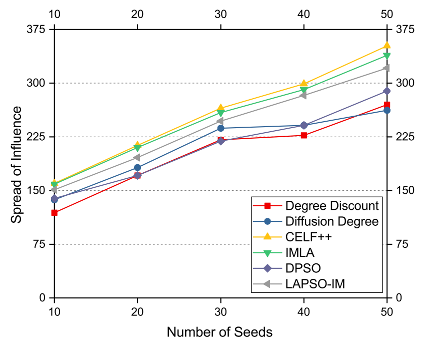
(c) Ca-GrQc



(d) NetHEPT

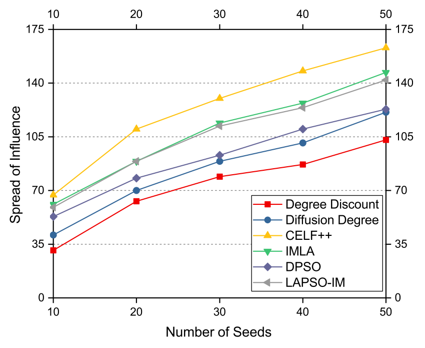


(e) Gnutella30

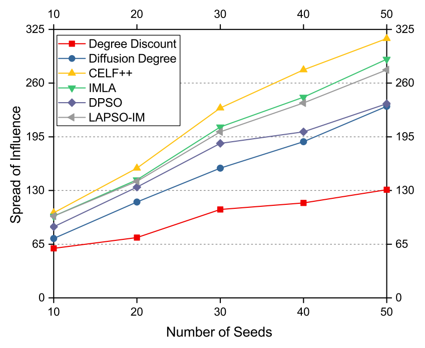


(f) NetPHY

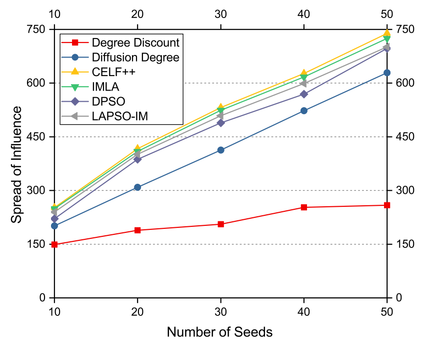
Fig. 10. Influence spread comparison in various datasets under IC model.



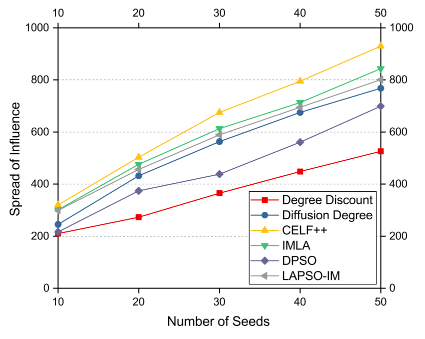
(a) NetInfective



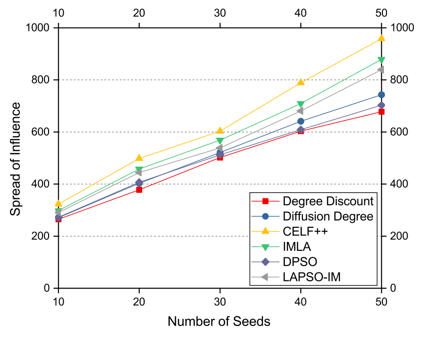
(b) NetScience



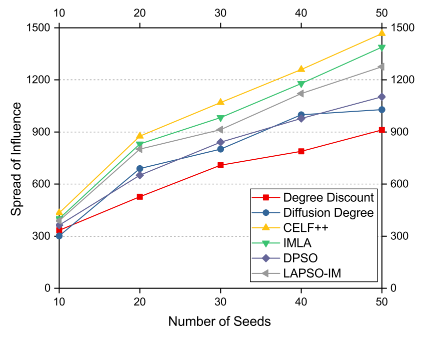
(c) Ca-GrQc



(d) NetHEPT



(e) Gnutella30

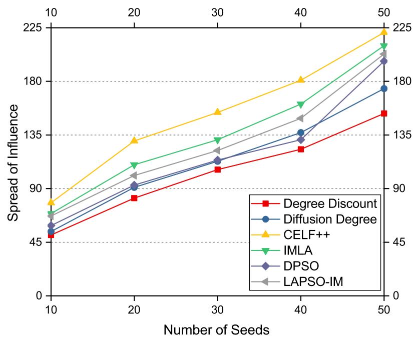


(f) NetPHY

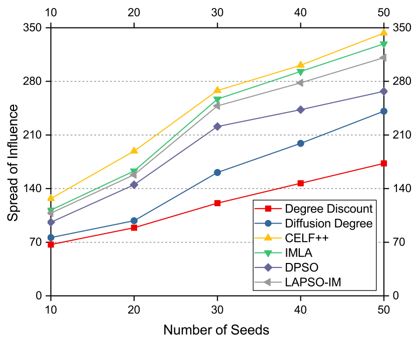
Fig. 11. Influence spread comparison in various datasets under WC model.

LAPSO-IM influence spread is slightly lower than IMLA and CELF++ under each diffusion models for all six real-world social networks. This is because of IMLA and CELF++ both have more stable search strategy with time-consuming MC simulations. Therefore, to acquire a trade-off between influence spread and time, we compromise with the quality of LAPSO-IM in very less to gain high efficiency.

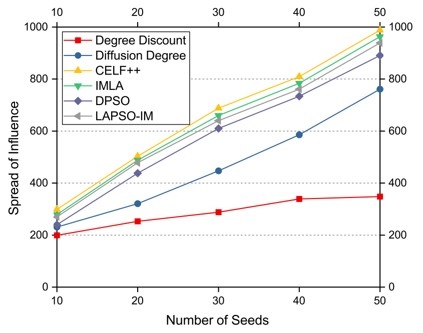
We summarize the influence spread of LAPSO-IM with the state-of-the-art algorithms on each dataset under traditional diffusion models at $S = 50$ as depicted in Tables 6, 7, and 8. The



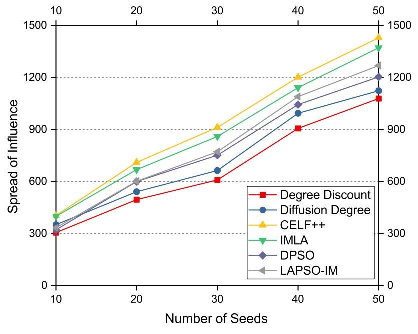
(a) NetInfective



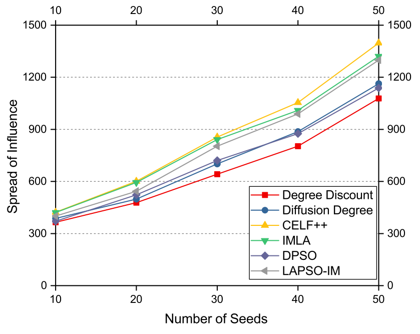
(b) NetScience



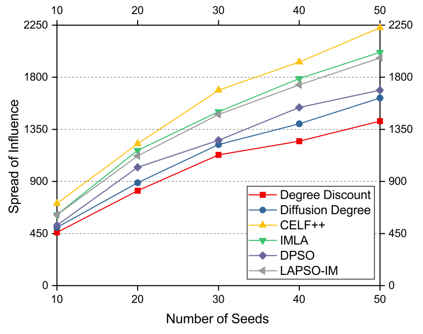
(c) Ca-GrQc



(d) NetHEPT



(e) Gnutella30



(f) NetPHY

Fig. 12. Influence spread comparison in various datasets under LT model.

influence spread is denoted as percentage of nodes in the original network which are influenced by seed set S. As shown in Tables 6, 7, and 8, the influence spread of LAPSO-IM is better than heuristic techniques and DPSO. The influence spread of LAPSO-IM is quite close to IMLA and CELF++.

6.4.2. *Efficiency*

In this section, we compare the running time of the proposed algorithm with the state-of-the-art algorithms to study how efficiently proposed algorithm finds the distinct size seed set on different datasets. For efficiency analysis, we perform the experiment on same size seed set as used in quality analysis of the same six real-world networks. Figs. 13, 14, and 15 show the comparison of LAPSO-IM with the best suited state-of-the-art algorithms under IC, WC, and LT diffusion models respectively.

As shown in Figs. 13, 14, and 15, both heuristic approaches Degree Discount and Diffusion Degree are much faster than LAPSO-IM under each diffusion models for all networks. This is because both heuristics consider only node degree to select seed nodes. Heuristic approaches take almost same running time. LAPSO-IM takes almost same running time as base method DPSO under each diffusion models for all social networks. LAPSO-IM outperforms both IMLA and CELF++ in terms of efficiency under each diffusion models for all networks. This is because IMLA and CELF++ both have time-consuming MC simulations to simulate influence spread. LAPSO-IM is more efficient than base method IMLA with approximate influence spread and is better than base method DPSO regarding influence spread with comparable running time. Therefore, LAPSO-IM acquires a trade-off between influence spread and running time.

In Tables 6, 7, and 8, we summarize the running time (in seconds) of LAPSO-IM with the state-of-the-art algorithms on each dataset under traditional diffusion models at $S = 50$. LAPSO-IM is faster than IMLA and CELF++ and the running time of LAPSO-IM is very approximate to base technique DPSO. Therefore, Tables 6, 7, and 8 show that LAPSO-IM acquires a trade-off between influence spread and time.

6.5. *Statistical test*

In this section, we present statistical tests to analyze the significant differences in influence spread between the proposed algorithm and compared algorithms. First, we applied the Friedman test [55,56] to analyze whether there are significant differences in the average influence spread. If there are significant differences, then we applied the Holland–Bonferroni procedure [56,57] and the Holm–Bonferroni procedure [56] as post hoc procedure to find the degree of rejection of each hypothesis. Both the post hoc procedures consider LAPSO-IM as control algorithm. We consider the level of confidence $\alpha_c = 0.05$ and degree of freedom $D_f = 5$ in all the cases.

The Friedman test indicates that there is a significant difference in average influence spread for each seed set under IC model as shown in Table 9. The null hypothesis (H_0) states that the methods compared are statistically equivalent, with no significant difference. The Friedman test rejects the hypothesis H_0 if the test statistic value F_f is greater than the $\chi^2(\alpha_c, D_f)$, i.e. $F_f > 11.0705$. Table 9 performs the Friedman test for average influence spread and indicates that the null hypothesis is rejected for each k. Similarly, the Friedman test also rejects the null hypothesis for each k under WC and LT models. Therefore, the Holland–Bonferroni procedure (independent test statistics) and the Holm–Bonferroni procedure (dependent test statistics) are applied to measure the concrete differences between the algorithms.

The Holland–Bonferroni procedure rejects hypothesis H_1 to H_{i-1} if p-value is greater than adjusted level of confidence value, i.e., $p_i > 1 - (1 - \alpha_c)^{D_f - i}$, where i is the smallest integer. We consider independent test statistics to perform this post hoc procedure. Tables 10, 11, and 12 show the p-value/i results for average influence spread under IC, WC, and LT diffusion models respectively. In the tables, "*" and highlighted values indicate the control algorithm and rejected hypothesis respectively.

The Holm–Bonferroni procedure rejects hypothesis H_1 to H_{i-1} if the p-value is greater than the adjusted level of confidence

Table 6
Comparison of influence spread (represented in terms of percentage of nodes in original social networks) and running time (in terms of seconds) under IC diffusion model in various datasets at $|S| = 50$.

Dataset	Influence spread						Running time					
	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM
NetInfective	26.83	27.32	34.88	32.44	27.56	30.00	0.02	0.04	2201.09	1598.35	24.19	24.28
NetScience	5.22	5.85	6.73	6.36	5.48	6.23	0.06	0.70	223.43	79.05	26.03	29.02
Ca-GrQc	1.18	1.22	1.30	1.28	1.24	1.25	0.12	0.14	1407.78	817.39	26.01	29.02
NetHEPT	0.64	0.68	0.85	0.77	0.70	0.74	0.35	0.38	3849.59	1835.87	37.94	41.89
Gnutella30	0.44	0.59	0.86	0.79	0.63	0.76	0.35	0.67	5182.30	2103.56	57.89	63.46
NetPHY	0.73	0.71	0.95	0.91	0.78	0.86	1.06	1.29	5487.20	2421.21	64.59	71.32

Table 7
Comparison of influence spread (represented in terms of percentage of nodes in original social networks) and running time (in terms of seconds) under WC diffusion model in various datasets at $|S| = 50$.

Dataset	Influence spread						Running time					
	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM
NetInfective	29.51	31.71	39.76	35.85	30.00	34.63	0.02	0.04	1901.00	1198.00	18.20	20.28
NetScience	8.24	14.60	19.76	18.19	14.79	17.36	0.06	0.70	171.40	69.10	21.00	24.02
Ca-GrQc	4.94	12.00	14.10	13.83	13.30	13.37	0.12	0.14	1208.00	717.00	24.00	27.02
NetHEPT	3.45	5.04	6.11	5.53	4.59	5.25	0.35	0.38	2850.00	1436.00	27.90	36.89
Gnutella30	1.85	2.03	2.61	2.40	1.92	2.28	0.35	0.67	4182.00	1704.00	47.90	53.46
NetPHY	2.45	2.77	3.95	3.74	2.97	3.43	1.06	1.29	4487.00	1921.00	57.60	64.32

Table 8
Comparison of influence spread (represented in terms of percentage of nodes in original social networks) and running time (in terms of seconds) under LT diffusion model in various datasets at $|S| = 50$.

Dataset	Influence spread						Running time					
	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM	Degree Discount	Diffusion Discount	CELF++	IMLA	DPSO	LAPSO-IM
NetInfective	37.32	42.44	53.90	51.22	48.05	49.58	0.02	0.04	1801.00	1108.00	16.20	20.28
NetScience	10.89	15.17	21.59	20.70	16.80	19.57	0.06	0.70	161.00	61.10	21.00	24.02
Ca-GrQc	6.64	14.52	18.85	18.37	17.00	17.91	0.12	0.14	1108.00	701.00	24.00	27.02
NetHEPT	7.08	7.37	9.38	9.00	7.90	8.32	0.35	0.38	2650.00	1336.00	27.90	36.89
Gnutella30	2.94	3.17	3.81	3.60	3.10	3.54	0.35	0.67	3782.00	1604.00	46.90	53.46
NetPHY	3.82	4.36	6.00	5.43	4.55	5.29	1.06	1.29	4287.00	1821.00	56.60	63.32

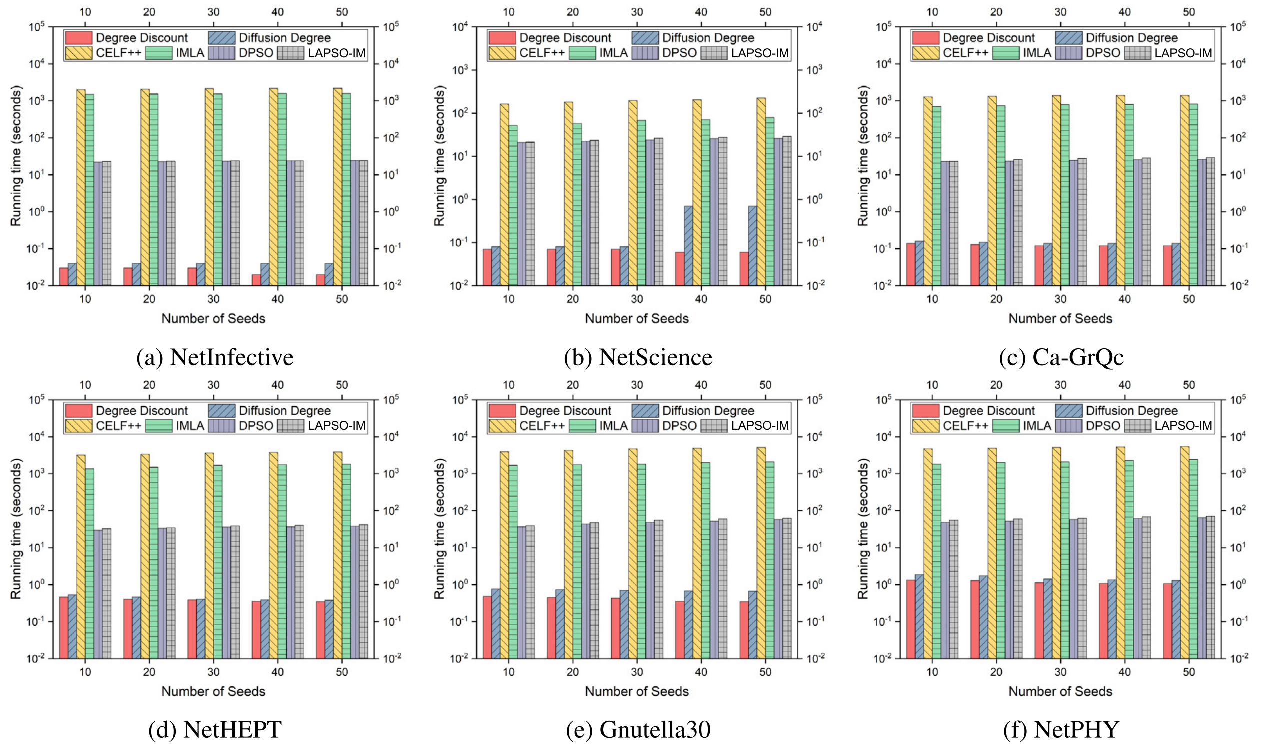
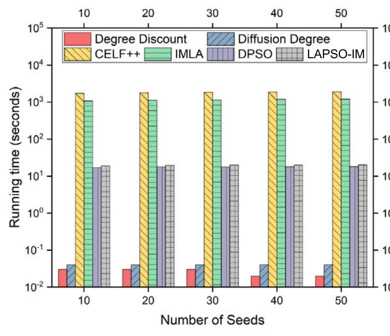
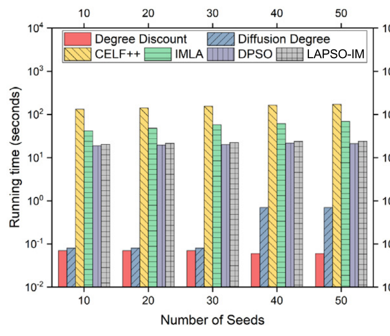


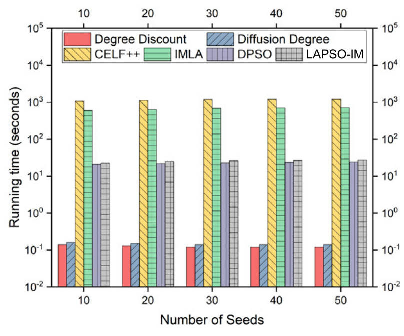
Fig. 13. The running time comparison in various datasets under IC model.



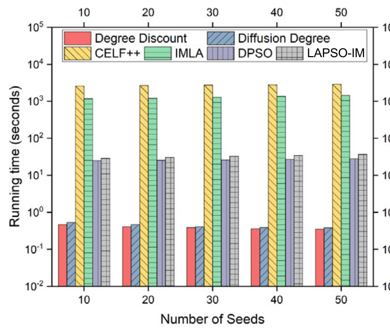
(a) NetInfective



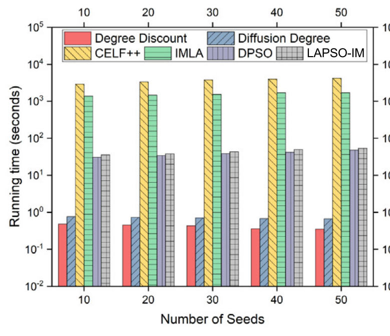
(b) NetScience



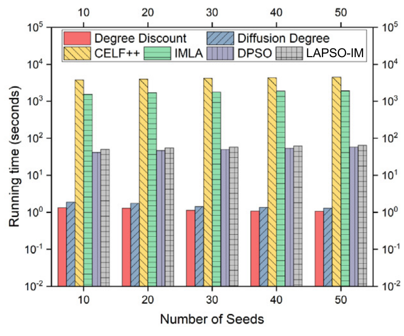
(c) Ca-GrQc



(d) NetHEPT

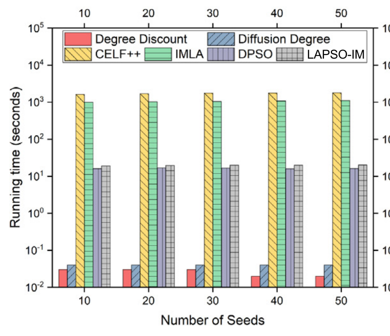


(e) Gnutella30

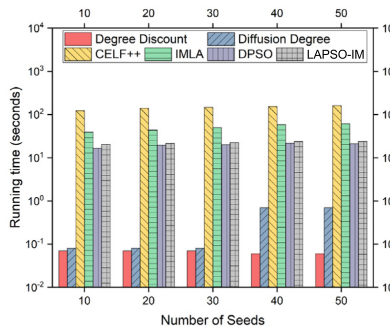


(f) NetPHY

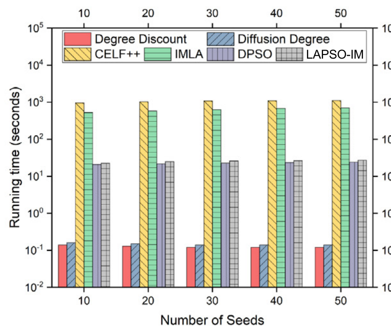
Fig. 14. The running time comparison in various datasets under WC model.



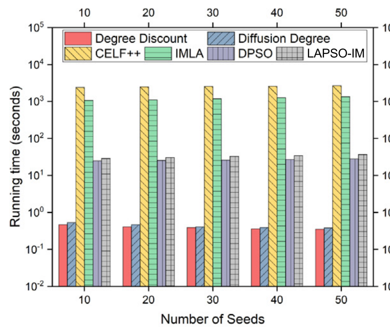
(a) NetInfective



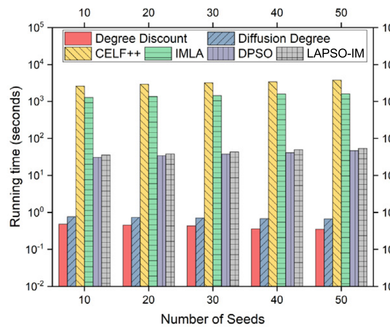
(b) NetScience



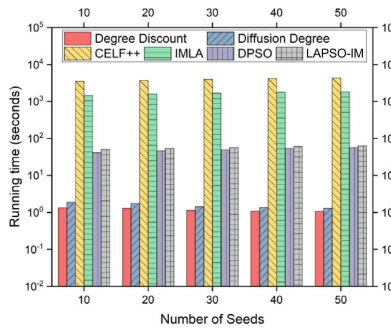
(c) Ca-GrQc



(d) NetHEPT



(e) Gnutella30



(f) NetPHY

Fig. 15. The running time comparison in various datasets under LT model.

value, i.e., $p_i > \alpha_c/((D_f+1)-i)$, where i is the smallest integer. We consider dependent test statistics to perform Holm–Bonferroni post hoc procedure. Let $p_1, p_2, \ldots, p_{D_f}$ are the ordered p-values (smallest to largest) and $H_1, H_2, \ldots, H_{D_f}$ are the corresponding hypothesis. The Holm–Bonferroni procedure starts with the most significant p-value. Tables 13, 14, and 15 show the p-value/i results for average influence spread under IC, WC, and LT diffusion models respectively.

Table 9
The Friedman test on average influence spread under IC diffusion model.

Seed	Dataset	IS-value/Rank						Test value	State result
		Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM	F_f	Is $F_f > \chi^2$?
10	NetInefective	42/1	47/2	63/6	59/5	51/3	58/4	22.3571	Null hypothesis rejected
	NetScience	29/1.5	32/5	35/6	31/3.5	29/1.5	31/3.5		
	GrQc	18/1.5	19/3.5	20/5.5	20/5.5	18/1.5	19/3.5		
	NetHEPT	33/1	35/2.5	41/6	39/5	35/2.5	38/4		
	Gnutella30	77/1	89/2	152/6	147/5	129/3	142/4		
	NetPHY	119/1	137/2	160/6	159/5	139/3	151/4		
20	NetInefective	67/1	75/2	91/6	85/5	82/3	83/4	21.5357	Null hypothesis rejected
	NetScience	37/1	51/2	60/6	54/5	46/3	53/4		
	GrQc	31/1	33/2	35/6	34/4.5	33/3	34/4.5		
	NetHEPT	56/1	63/3.5	70/6	65/5	59/2	63/3.5		
	Gnutella30	108/1	143/2	202/6	187/5	149/3	179/4		
	NetPHY	171/1	182/3	213/6	210/5	171/2	196/4		
30	NetInefective	76/1	89/2.5	117/6	107/5	89/2.5	99/4	28.9047	Null hypothesis rejected
	NetScience	58/1	71/3	86/6	78/5	67/2	75/4		
	GrQc	41/1	47/2	51/6	50/4.5	48/3	50/4.5		
	NetHEPT	78/1	86/3	94/6	87/5	85/2	86/4		
	Gnutella30	131/1	169/2	243/6	214/5	178/3	203/4		
	NetPHY	221/1	237/3	265/6	259/5	229/2	247/4		
40	NetInefective	93/1	99/2	131/6	119/5	101/3	108/4	29.5238	Null hypothesis rejected
	NetScience	71/1	78/2	96/6	89/5	79/3	86/4		
	GrQc	52/1	56/2.5	60/6	59/5	56/2.5	58/4		
	NetHEPT	89/1	95/2	111/6	103/5	101/3	102/4		
	Gnutella30	147/1	205/2	278/6	241/5	199/3	236/4		
	NetPHY	227/1	241/2.5	299/6	291/5	241/2.5	283/4		
50	NetInefective	110/1	112/2	143/6	133/5	113/3	123/4	29.5238	Null hypothesis rejected
	NetScience	83/1	93/3	107/6	101/5	87/2	99/4		
	GrQc	62/1	64/2	68/6	67/5	65/3	66/4		
	NetHEPT	97/1	103/2	129/6	118/5	106/3	114/4		
	Gnutella30	163/1	216/2	317/6	289/5	231/3	279/4		
	NetPHY	270/1	262/2	352/6	339/5	289/3	321/4		

Table 10
The estimation of p-value/i based on the Holland procedure (Independent test) for post hoc analysis on average influence spread under IC diffusion model.

Seed	FRank/z-score/*p*-value/i					
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM
10	1.16/2.4719/0.01344/1	**2.83/0.9258/0.35455/4.5**	5.91/1.9257/0.05414/2	**4.83/0.9258/0.35455/4.5**	**2.41/1.3146/0.18864/3**	3.83/0/1/*
20	1/2.1571/0.03099/2	**2.41/0.8517/0.39438/4**	6/2.4997/0.01243/1	5/1.5461/0.12208/3	**2.66/0.6203/0.53506/5**	3.33/0/1/*
30	1/2.8515/0.00435/1	**2.58/1.3887/0.16492/4**	6/1.7758/0.07576/2	**4.91/0.7684/0.44225/5**	2.41/1.5461/0.12208/3	4.08/0/1/*
40	1/2.7774/0.00548/1	2.16/1.7033/0.08851/3	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	**2.83/1.0832/0.27872/4**	4/0/1/*
50	1/2.7774/0.00548/1	2.16/1.7033/0.08851/3	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	**2.83/1.0832/0.27872/4**	4/0/1/*

Table 11
The estimation of p-value/i based on the Holland procedure (Independent test) for post hoc analysis on average influence spread under WC diffusion model.

Seed	FRank/z-score/*p*-value/i					
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM
10	1/2.8515/0.00435/1	2.25/1.6942/0.09022/3	6/1.7758/0.07576/2	**4.91/0.7684/0.44225/5**	**2.75/1.2313/0.21821/4**	4.08/0/1/*
20	1/2.8515/0.00435/1	2.25/1.6942/0.09022/3	6/1.7758/0.07576/2	**4.91/0.7684/0.44225/5**	**2.75/1.2313/0.21821/4**	4.08/0/1/*
30	1/2.7774/0.00548/1	2.33/1.5461/0.12208/3	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	**2.66/1.2406/0.21475/4**	4/0/1/*
40	1/2.7774/0.00548/1	**2.5/1.3887/0.16492/4**	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	2.33/1.5461/0.12208/3	4/0/1/*
50	1/2.7774/0.00548/1	**2.5/1.3887/0.16492/3.5**	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	**2.5/1.3887/0.16492/3.5**	4/0/1/*

Table 12
The estimation of p-value/i based on the Holland procedure (Independent test) for post hoc analysis on average influence spread under LT diffusion model.

Seed	FRank/z-score/*p*-value/i					
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM
10	1/2.7774/0.00548/1	2.16/1.7033/0.08851/3	6/1.7758/0.07576/2	**5/0.9258/0.35455/5**	**2.83/1.0832/0.27872/4**	4/0/1/*
20	1/2.7774/0.00548/1	2/1.8516/0.06408/2	6/1.7758/0.07576/3	5/0.9258/0.35455/4.5	**3/0.9258/0.35455/4.5**	4/0/1/*
30	1/2.7774/0.00548/1	2/1.8516/0.06408/2.5	6/1.8516/0.06408/2.5	**5/0.9258/0.35455/4.5**	**3/0.9258/0.35455/4.5**	4/0/1/*
40	1/2.7774/0.00548/1	2.33/1.5461/0.12208/3	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	**2.66/1.2406/0.21475/4**	4/0/1/*
50	1/2.7774/0.00548/1	2.16/1.7033/0.08851/3	6/1.8516/0.06408/2	**5/0.9258/0.35455/5**	**2.83/1.0832/0.27872/4**	4/0/1/*

Table 13
The estimation of p-value/i based on the Holm procedure (Dependent test) for post hoc analysis on average influence spread under IC diffusion model.

Seed	FRank/z-score/*p*-value/i					
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM
10	1.16/2.4719/0.01344/1	2.83/0.9258/0.35455/4.5	5.91/1.9257/0.05414/2	4.83/0.9258/0.35455/4.5	2.41/1.3146/0.18864/3	3.83/0/1/*
20	1/2.1571/0.03099/2	2.41/0.8517/0.39438/4	6/2.4997/0.01243/1	5/1.5461/0.12208/3	2.66/0.6203/0.53506/5	3.33/0/1/*
30	**1/2.8515/0.00435/1**	2.58/1.3887/0.16492/4	6/1.7758/0.07576/2	4.91/0.7684/0.44225/5	2.41/1.5461/0.12208/3	4.08/0/1/*
40	**1/2.7774/0.00548/1**	2.16/1.7033/0.08851/3	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.83/1.0832/0.27872/4	4/0/1/*
50	**1/2.7774/0.00548/1**	2.16/1.7033/0.08851/3	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.83/1.0832/0.27872/4	4/0/1/*

Table 14
The estimation of p-value/i based on the Holm procedure (Dependent test) for post hoc analysis on average influence spread under WC diffusion model.

Seed	FRank/z-score/*p*-value/i					
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM
10	**1/2.8515/0.00435/1**	2.25/1.6942/0.09022/3	6/1.7758/0.07576/2	4.91/0.7684/0.44225/5	2.75/1.2313/0.21821/4	4.08/0/1/*
20	**1/2.8515/0.00435/1**	2.25/1.6942/0.09022/3	6/1.7758/0.07576/2	4.91/0.7684/0.44225/5	2.75/1.2313/0.21821/4	4.08/0/1/*
30	**1/2.7774/0.00548/1**	2.33/1.5461/0.12208/3	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.66/1.2406/0.21475/4	4/0/1/*
40	**1/2.7774/0.00548/1**	2.5/1.3887/0.16492/4	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.33/1.5461/0.12208/3	4/0/1/*
50	**1/2.7774/0.00548/1**	2.5/1.3887/0.16492/3.5	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.5/1.3887/0.16492/3.5	4/0/1/*

Table 15
The estimation of p-value/i based on the Holm procedure (Dependent test) for post hoc analysis on average influence spread under LT diffusion model.

Seed	FRank/z-score/*p*-value/i					
	Degree Discount	Diffusion Degree	CELF++	IMLA	DPSO	LAPSO-IM
10	**1/2.7774/0.00548/1**	2.16/1.7033/0.08851/3	6/1.7758/0.07576/2	5/0.9258/0.35455/5	2.83/1.0832/0.27872/4	4/0/1/*
20	**1/2.7774/0.00548/1**	2/1.8516/0.06408/2	6/1.7758/0.07576/3	5/0.9258/0.35455/4.5	3/0.9258/0.35455/4.5	4/0/1/*
30	**1/2.7774/0.00548/1**	2/1.8516/0.06408/2.5	6/1.8516/0.06408/2.5	5/0.9258/0.35455/4.5	3/0.9258/0.35455/4.5	4/0/1/*
40	**1/2.7774/0.00548/1**	2.33/1.5461/0.12208/3	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.66/1.2406/0.21475/4	4/0/1/*
50	**1/2.7774/0.00548/1**	2.16/1.7033/0.08851/3	6/1.8516/0.06408/2	5/0.9258/0.35455/5	2.83/1.0832/0.27872/4	4/0/1/*

The statistical tests on the IC, WC and LT models demonstrate LAPSO-IM is comparable to both IMLA and CELF++ algorithm, and significantly superior to the remaining three algorithms in terms of influence spread. However, LAPSO-IM outperforms both IMLA and CELF++ in terms of efficiency. Therefore, the proposed algorithm acquires a trade-off between influence spread and efficiency.

7. Conclusion and future directions

In this paper, we have presented a learning automata based discrete particle swarm optimization approach to maximize influence in social networks. First, we have extended local influence evaluation function *expected diffusion value* to approximate within two-hope area. Next, we have presented LAPSO-IM to optimize local influence evaluation function by redefining the update rule of velocity and position vectors for each action of learning automata. The experimental results on six real-world networks show that the proposed algorithm performs better than DPSO regarding influence spread with almost the same running time. The proposed algorithm outperforms both heuristics Degree Discount and Diffusion Degree in terms of influence spread with low efficiency. LAPSO-IM is more time efficient than both IMLA and CELF++ with approximate influence spread. The influence spread of LAPSO-IM is better than DPSO with comparable efficiency. Therefore, the LAPSO-IM algorithm is a trade-off between effectiveness and efficiency. LAPSO-IM can obtain a good performance with low time-consuming.

Much future technical work is possible, for example, we can extend our proposed algorithm into multi-objective IM optimization problem with some constraints such as competitive, spatial, topic, etc., to provide more trade-off decisions in the more realistic and practical scenario. To optimize noisy fitness, we will adopt learning automata induced artificial bee colony optimization. Furthermore, we will move towards IM across social networks because social networks are coupled and interdependent with each other, and not self-governed.

8. Compliance with ethical standards

The article does not contain any studies with human or animal subjects. This article presents an algorithm LAPSO-IM.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105554.

References

[1] J. Goldenberg, B. Libai, E. Muller, Talk of the network: A complex systems look at the underlying process of word-of-mouth, Mark. Lett. 12 (3) (2001) 211–223, http://dx.doi.org/10.1023/A:1011122126881.

[2] P. Domingos, M. Richardson, Mining the network value of customers, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '01, ACM, New York, NY, USA, 2001, pp. 57–66, http://dx.doi.org/10.1145/502512.502525.

[3] P. Wu, L. Pan, Scalable influence blocking maximization in social networks under competitive independent cascade models, Comput. Netw. 123 (2017) 38–50, http://dx.doi.org/10.1016/j.comnet.2017.05.004.

[4] W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in: Proceedings of the 2010 IEEE International Conference on Data Mining, in: ICDM '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 88–97, http://dx.doi.org/10.1109/ICDM.2010.118.

[5] M. Ye, X. Liu, W.-C. Lee, Exploring social influence for recommendation: A generative model approach, in: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '12, ACM, New York, NY, USA, 2012, pp. 671–680, http://dx.doi.org/10.1145/2348283.2348373.

[6] Y.-W. Teng, C.-H. Tai, P.S. Yu, M.-S. Chen, Revenue maximization on the multi-grade product, in: Proceedings of the 2018 SIAM International Conference on Data Mining, 2018, pp. 576–584, http://dx.doi.org/10.1137/1.9781611975321.65.

[7] J. Leskovec, L.A. Adamic, B.A. Huberman, The dynamics of viral marketing, ACM Trans. Web 1 (1) (2007) http://dx.doi.org/10.1145/1232722.1232727.

[8] D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '03, ACM, New York, NY, USA, 2003, pp. 137–146, http://dx.doi.org/10.1145/956750.956769.

[9] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '07, ACM, New York, NY, USA, 2007, pp. 420–429, http://dx.doi.org/10.1145/1281192.1281239.

[10] Y. Wang, X. Feng, A potential-based node selection strategy for influence maximization in a social network, in: R. Huang, Q. Yang, J. Pei, J. Gama, X. Meng, X. Li (Eds.), Advanced Data Mining and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 350–361.

[11] Y. Wang, G. Cong, G. Song, K. Xie, Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, in: KDD, 2010.

[12] A. Goyal, W. Lu, L.V. Lakshmanan, Celf++: Optimizing the greedy algorithm for influence maximization in social networks, in: Proceedings of the 20th International Conference Companion on World Wide Web, in: WWW '11, ACM, New York, NY, USA, 2011, pp. 47–48, http://dx.doi.org/10.1145/1963192.1963217.

[13] S.S. Singh, K. Singh, A. Kumar, B. Biswas, Coim: Community-based influence maximization in social networks, in: A.K. Luhach, D. Singh, P.-A. Hsiung, K.B.G. Hawari, P. Lingras, P.K. Singh (Eds.), Advanced Informatics for Computing Research, Springer Singapore, Singapore, 2019, pp. 440–453.

[14] M. Kimura, K. Saito, Tractable models for information diffusion in social networks, in: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Knowledge Discovery in Databases: PKDD 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 259–271.

[15] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '09, ACM, New York, NY, USA, 2009, pp. 199–208, http://dx.doi.org/10.1145/1557019.1557047.

[16] W. Chen, C. Wang, Y. Wang, Scalable influence maximization for prevalent viral marketing in large-scale social networks, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '10, ACM, New York, NY, USA, 2010, pp. 1029–1038, http://dx.doi.org/10.1145/1835804.1835934.

[17] W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in: 2010 IEEE International Conference on Data Mining, 2010, pp. 88–97, http://dx.doi.org/10.1109/ICDM.2010.118.

[18] S.S. Singh, A. Kumar, K. Singh, B. Biswas, C2im: Community based context-aware influence maximization in social networks, Physica A 514 (2019) 796–818, http://dx.doi.org/10.1016/j.physa.2018.09.142, URL http://www.sciencedirect.com/science/article/pii/S0378437118312822.

[19] S. Cheng, H. Shen, J. Huang, G. Zhang, X. Cheng, Staticgreedy: Solving the scalability-accuracy dilemma in influence maximization, in: Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, in: CIKM '13, ACM, New York, NY, USA, 2013, pp. 509–518, http://dx.doi.org/10.1145/2505515.2505541, URL http://doi.acm.org/10.1145/2505515.2505541.

[20] N. Ohsaka, T. Akiba, Y. Yoshida, K.-I. Kawarabayashi, Fast and accurate influence maximization on large networks with pruned monte-carlo simulations, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, in: AAAI'14, AAAI Press, 2014, pp. 138–144, URL http://dl.acm.org/citation.cfm?id=2893873.2893897.

[21] Y. Tang, X. Xiao, Y. Shi, Influence maximization: near-optimal time complexity meets practical efficiency, in: SIGMOD Conference, 2014.

[22] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on, 1995, pp. 39–43, http://dx.doi.org/10.1109/MHS.1995.494215.

[23] M. Hasanzadeh, M.R. Meybodi, S.S. Ghidary, Improving learning automata based particle swarm: An optimization algorithm, in: 2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI), 2011, pp. 291–296, http://dx.doi.org/10.1109/CINTI.2011.6108517.

[24] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, K. Xie, Simulated annealing based influence maximization in social networks, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, in: AAAI'11, AAAI Press, 2011, pp. 127–132, URL http://dl.acm.org/citation.cfm?id=2900423.2900443.

[25] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, Oper. Res. Lett. 32 (1) (2004) 41–43, http://dx.doi.org/10.1016/S0167-6377(03)00062-2.

[26] H. Ge, J. Huang, C. Di, J. Li, S. Li, Learning automata based approach for influence maximization problem on social networks, in: 2017 IEEE Second International Conference on Data Science in Cyberspace (DSC), 2017, pp. 108–117, http://dx.doi.org/10.1109/DSC.2017.54.

[27] S. Kundu, C.A. Murthy, S.K. Pal, A new centrality measure for influence maximization in social networks, in: S.O. Kuznetsov, D.P. Mandal, M.K. Kundu, S.K. Pal (Eds.), Pattern Recognition and Machine Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 242–247.

[28] J. Kim, S.K. Kim, H. Yu, Scalable and parallelizable processing of influence maximization for large-scale social networks?, in: 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 266–277, http://dx.doi.org/10.1109/ICDE.2013.6544831.

[29] Y.-Y. Ko, K.-J. Cho, S.-W. Kim, Efficient and effective influence maximization in social networks: A hybrid-approach, Inform. Sci. 465 (2018) 144–161, http://dx.doi.org/10.1016/j.ins.2018.07.003, URL http://www.sciencedirect.com/science/article/pii/S0020025518305176.

[30] A. Goyal, W. Lu, L.V.S. Lakshmanan, Simpath: An efficient algorithm for influence maximization under the linear threshold model, in: Proceedings of the 2011 IEEE 11th International Conference on Data Mining, in: ICDM '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 211–220, http://dx.doi.org/10.1109/ICDM.2011.132.

[31] K. Jung, W. Heo, W. Chen, Irie: Scalable and robust influence maximization in social networks, in: Proceedings of the 2012 IEEE 12th International Conference on Data Mining, in: ICDM '12, IEEE Computer Society, Washington, DC, USA, 2012, pp. 918–923, http://dx.doi.org/10.1109/ICDM.2012.79.

[32] W.-S. Yang, S.-X. Weng, Application of the ant colony optimization algorithm to the influence-maximization problem, in: International Journal of Swarm Intelligence and Evolutionary Computation, Vol. 1, 2012.

[33] M. Gong, J. Yan, B. Shen, L. Ma, Q. Cai, Influence maximization in social networks based on discrete particle swarm optimization, Inform. Sci. 367–368 (2016) 600–614, http://dx.doi.org/10.1016/j.ins.2016.07.012, URL http://www.sciencedirect.com/science/article/pii/S002002551630500X.

[34] X. Zhu, Z. Wang, Y. Yang, B. Zhou, Y. Jia, Influence efficiency maximization: How can we spread information efficiently? J. Comput. Sci. 28 (2018) 245–256, http://dx.doi.org/10.1016/j.jocs.2017.11.001, URL http://www.sciencedirect.com/science/article/pii/S1877750317305288.

[35] N. Ohsaka, Y. Yamaguchi, N. Kakimura, K.-i. Kawarabayashi, Maximizing time-decaying influence in social networks, in: P. Frasconi, N. Landwehr, G. Manco, J. Vreeken (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, 2016, pp. 132–147.

[36] J. Fan, J. Qiu, Y. Li, Q. Meng, D. Zhang, G. Li, K. Tan, X. Du, OCTOPUS: an online topic-aware influence analysis system for social networks, in: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018, 2018, pp. 1569–1572, http://dx.doi.org/10.1109/ICDE.2018.00178, URL http://doi.ieeecomputersociety.org/10.1109/ICDE.2018.00178.

[37] W. Lu, W. Chen, L.V.S. Lakshmanan, From competition to complementarity: Comparative influence diffusion and maximization, CoRR abs/1507.00317. arXiv:1507.00317. URL http://arxiv.org/abs/1507.00317, 2015.

[38] L. Guo, D. Zhang, G. Cong, W. Wu, K. Tan, Influence maximization in trajectory databases, IEEE Trans. Knowl. Data Eng. 29 (3) (2017) 627–641, http://dx.doi.org/10.1109/TKDE.2016.2621038.

[39] J. Zhang, L. Xu, J. Ma, M. Zhou, A learning automata-based particle swarm optimization algorithm for noisy environment, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 141–147, http://dx.doi.org/10.1109/CEC.2015.7256885.

[40] P. Rakshit, A. Konar, A.K. Nagar, Learning automata induced artificial bee colony for noisy optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 984–991, http://dx.doi.org/10.1109/CEC.2017.7969415.

[41] S.S. Singh, K. Singh, A. Kumar, H.K. Shakya, B. Biswas, A survey on information diffusion models in social networks, in: A.K. Luhach, D. Singh, P.-A. Hsiung, K.B.G. Hawari, P. Lingras, P.K. Singh (Eds.), Advanced Informatics for Computing Research, Springer Singapore, Singapore, 2019, pp. 426–439.

[42] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proc. of Conf. on System, Man, and Cybernetics, 4104–4109, 1997.

[43] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), 1998, pp. 69–73, http://dx.doi.org/10.1109/ICEC.1998.699146.

[44] R.C. Eberhart, Y. Shi, Guest editorial special issue on particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 201–203, http://dx.doi.org/10.1109/TEVC.2004.830335.

[45] K.S. Narendra, M.A.L. Thathachar, Learning Automata: An Introduction, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[46] A. Hashemi, M. Meybodi, A note on the learning automata based algorithms for adaptive parameter selection in pso, Appl. Soft Comput. 11 (1) (2011) 689–705, http://dx.doi.org/10.1016/j.asoc.2009.12.030, URL http://www.sciencedirect.com/science/article/pii/S1568494609002877.

[47] S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L.V.S. Lakshmanan, M. Schmidt, Diffusion independent semi-bandit influence maximization, CoRR abs/1703.00557. arXiv:1703.00557. URL http://arxiv.org/abs/1703.00557, 2017.

[48] N.A. Christakis, J.H. Fowler, Connected: The Surprising Power of Our Social Networks and How They Shape Our Lives, Little, Brown, New York, USA, 2009.

[49] S. Pei, L. Muchnik, J.S. Andrade Jr, Z. Zheng, H.A. Makse, Searching for superspreaders of information in real-world social media, Sci. Rep. 4 (2014) 5547.

[50] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. cois Pinton, W.V. den Broeck, What's in a crowd? analysis of face-to-face behavioral networks, J. Theoret. Biol. 271 (1) (2011) 166–180, http://dx.doi.org/10.1016/j.jtbi.2010.11.033, URL http://www.sciencedirect.com/science/article/pii/S0022519310006284.

[51] M. Ripeanu, A. Iamnitchi, I. Foster, Mapping the gnutella network, IEEE Internet Comput. 6 (1) (2002) 50–57, http://dx.doi.org/10.1109/4236.978369.

[52] M. E.J.N.ewman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (2006) 036104.

[53] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Trans. Knowl. Discov. Data 1 (1) (2007) http://dx.doi.org/10.1145/1217299.1217301, URL http://doi.acm.org/10.1145/1217299.1217301.

[54] W. Chen, L.V.S. Lakshmanan, C. Castillo, Information and Influence Propagation in Social Networks, Morgan & Claypool Publishers, 2013.

[55] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Amer. Statist. Assoc. 32 (200) (1937) 675–701, http://dx.doi.org/10.1080/01621459.1937.10503522, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1937.10503522. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522.

[56] J. n Derrac, S.G. a, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18, http://dx.doi.org/10.1016/j.swevo.2011.02.002, URL http://www.sciencedirect.com/science/article/pii/S2210650211000034.

[57] B.S. Holland, M.D. Copenhaver, An improved sequentially rejective bonferroni test procedure, Biometrics 43 (2) (1987) 417–423, URL http://www.jstor.org/stable/2531823.

Shashank Sheshar Singh received M.Tech. degree in Computer Science and Engineering from Indian Institute of Technology, Roorkee (IITR). He received B.Tech. degree in Computer Science and Engineering from Kali Charan Nigam Institute of Technology (KCNIT), Banda affiliated to GBTU University, Lucknow. He is working toward the Ph.D. in Computer Science and Engineering from Indian Institute of Technology (BHU), Varanasi. His research interests include Data Mining, Influence Maximization, Link Prediction and Social Network Analysis.

Ajay Kumar completed his master of technology in Computer Science & Engineering from Samrat Ashok Technological Institute Vidisha (M.P.) and Bachelor of Technology in Computer Science & Engineering from R.K.D.F Institute of Science and Technology Bhopal (M.P.). He is pursuing Ph.D. in Computer Science and Engineering from Indian Institute of Technology (BHU), Varanasi. His research interests include Link Prediction and Influence Maximization in social/complex networks.

Kuldeep Singh is currently pursuing his Ph.D in Computer science & Engineering from Indian Institute of Technology (BHU) Varanasi. His research interest includes High utility itemsets mining, social network analysis, and data mining. He received his M.Tech degree in Computer science and Engineering from Guru Jambheshwar University of Science and Technology, Hisar (Haryana). He has 8 years of teaching and research experience.

Bhaskar Biswas received Ph.D. in Computer Science and Engineering from Indian Institute of Technology (BHU), Varanasi. He received the B.Tech. degree in Computer Science and Engineering from Birla Institute of Technology, Mesra. He is working as Associate Professor at Indian Institute of Technology (BHU), Varanasi in the Computer Science and Engineering department. His research interests include Data Mining,Text Analysis, Machine Learning, Influence Maximization, Link Prediction, Social Network Analysis.