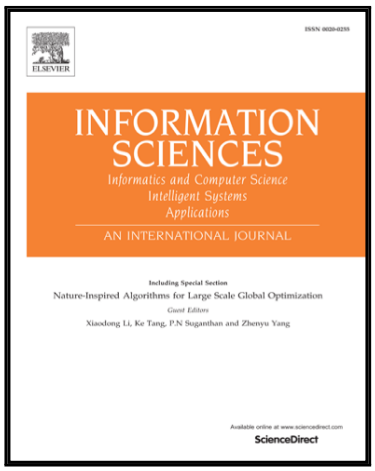


Influence Maximization in Social Networks Based on Discrete Particle Swarm Optimization

Maoguo Gong, Jianan Yan, Bo Shen, Lijia Ma, Qing Cai

PII: S0020-0255(16)30500-X
DOI: 10.1016/j.ins.2016.07.012
Reference: INS 12338

To appear in: *Information Sciences*

Received date: 10 April 2015
Revised date: 30 June 2016
Accepted date: 6 July 2016

Please cite this article as: Maoguo Gong, Jianan Yan, Bo Shen, Lijia Ma, Qing Cai, Influence Maximization in Social Networks Based on Discrete Particle Swarm Optimization, *Information Sciences* (2016), doi: 10.1016/j.ins.2016.07.012

Influence Maximization in Social Networks Based on Discrete Particle Swarm Optimization

Maoguo Gong[a,*], Jianan Yan[a], Bo Shen[a], Lijia Ma[a], Qing Cai[a]

[a] *Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an, Shaanxi Province 710071, China*

Abstract

Influence maximization in social networks aims to find a small group of individuals, which have maximal influence cascades. In this study, an optimization model based on a local influence criterion is established for the influence maximization problem. The local influence criterion can provide a reliable estimation for the influence propagations in independent and weighted cascade models. A discrete particle swarm optimization algorithm is then proposed to optimize the local influence criterion. The representations and update rules for the particles are redefined in the proposed algorithm. Moreover, a degree based heuristic initialization strategy and a network-specific local search strategy are introduced to speed up the convergence. Experimental results on four real-world social networks demonstrate the effectiveness and efficiency of the proposed algorithm for influence maximization.

Keywords: Social networks, Influence maximization, Cascade model, Particle swarm optimization

1. Introduction

With the great popularity of the Internet, plenty of online social networks, such as Facebook, Twitter and microblogs, have flourished in modern society. Having more than millions of users and billions of interconnections, social networks are not only serving as communication tools for users, but also potential marketing platforms for companies and advertisers [3, 13, 45]. Recent researches have shown that people place more value on recommendations from their friends than those from other channels like newspaper, TV and billboard [9, 59]. Thus, the effects of word-of-mouth, deriving from those close social circles, can make the information (i.e. interests, opinions or ideas) spread to get a wide range of cascading influence easily. Besides, compared with the traditional marketing techniques, marketing on social platforms has high profit and less investment. In recent years, many companies have chosen online social networks as marketing media to promote their new products, services or innovations. Due to limited budgets, companies are expected to select a small group of individuals as a seed set to make as many cascades as possible for the promotion [52, 57], a phenomenon called viral marketing. Influence maximization (IM) is to find a set of individuals that can generate maximum influence spread in complex networks [34, 62].

Inspired by the marketing promotion, Domingos and Richardson investigated the influence maximization as an optimization problem [15]. Kempe et al. [30, 31] pointed out that IM is essentially an NP-hard optimization problem and they proposed a greedy algorithm to solve the problem. They proved that the greedy method can guarantee $(1-1/e)$ (where e is the base of natural logarithms) optimal influence spread with respect to three commonly used information spread models, i.e., the independent cascade (IC) model [8, 20], the weighted cascade (WC) model [10, 31] and the linear threshold (LT) model [24, 48]. In order to determine an initial set with good performance in influence spread, the greedy method traverses all of nodes in the targeted social network. Besides, the computation of the influence spread for a given set is a #P-hard problem [35, 61] and the greedy algorithm combined with Monte Carlo simulations needs to run many times to obtain a precise result. The issues above result in the inefficiency of the greedy algorithm [9, 11] on the influence maximization.

*Corresponding author
Email address: gong@ieee.org (Maoguo Gong)

In recent years, many studies have been proposed to enhance the efficiency of the greedy algorithm. Leskovec et al. [37] put forward an improved greedy method by introducing a "Cost-Efficient Lazy Forward" (CELF) scheme. The CELF method can speed up the greedy algorithm by 700 times almost. Then Chen et al. [10] developed the NewGreedy and MixedGreedy methods to improve the greedy algorithm in different ways. Moreover, a community-based greedy method was devised by Wang et al. [53] to improve the greedy method, using the priori knowledge of the community structures of networks [42, 54]. CELF++ proposed by Goyal et al. optimized CELF by exploiting submodularity and experiments show that CELF++ is 35% $\sim$ 55% faster than CELF [23, 31]. And Kundu and Pal [34] proposed a deprecation based greedy strategy by estimating the performance of nodes and marking the nodes to be deprecated, and they have proved that the method can provide a guaranteed target node set when the influence function is monotonic and sub-modular. In recent studies, He and Kempe [25] focused on the effect of noise on the performance of influence maximization. They pointed that optimizing the submodularity may obtain highly suboptimal solutions, because the noise in the social networks may result in that the submodularity does not hold in influence maximization. Moreover, Morone and Makse [40] transformed the influence maximization problem into the optimal percolation in random networks to identify the initial seed set, which is indeed a many-body problem with crucial topological interactions between them [1].

Some heuristic approaches, including degree based and other centrality based methods [4, 30, 50], select the top-k influential nodes according to the property of each single node. Those methods may not consider the influence overlapping problem. Jiang et al. [29] introduced a local approximation, called the expected diffusion value (EDV), for the influence spread in the IC model to approximate the influence spread calculated by the Monte Carlo simulations. Besides, a simulated annealing based method (SAEDV) has been adopted to optimize the EDV and it can obtain good results with millions of iterations. However, the proposed EDV assumes that the propagation possibility in the IC model is very small. When the propagation possibility is high, it cannot approximate the influence spread very well.

Particle swarm optimization (PSO), as one of the swarm intelligence algorithms, was first proposed by Kennedy and Eberhart [16, 32] in 1995. PSO is a population-based algorithm inspired from the behaviors of bird flocks and fish schools. The individuals in PSO are called particles and each particle has a position and velocity vector. The position of a particle is viewed as a candidate solution for the target optimization problem. Owing to its simplicity and efficiency, PSO has become an important tool to solve optimization problems [38, 55]. Over the past decades, it has been successfully applied to solve various optimization problems, especially for those intractable NP-hard issues [7, 22].

In this paper, we propose a discrete particle swarm optimization (DPSO) algorithm for the influence maximization in social networks. Our main contributions are listed as follows:

1) A fast local influence estimation (LIE) function is introduced to approximate the two-hop influence of initial set. Consequently, the influence maximization is transformed into the optimization of the LIE function.

2) A DPSO algorithm is put forward to optimize the LIE function by redefining the representations and update rules for position and velocity. A degree based heuristic initialization method and a network-specific local search strategy are presented to make the proposed algorithm converge quickly.

3) Experimental results on four real-world social networks show that the proposed DPSO algorithm not only performs comparatively to the state-of-the-art CELF++ greedy method, but also achieves much lower running time in both the IC and WC spread models.

The rest of this paper is organized as follows. Section 2 presents the related works. Section 3 gives the detailed descriptions of the proposed algorithm. Sections 4 and 5 show the experimental results and the concluding remarks, respectively.

2. Related Works

A social network can be modeled as a graph $G = (N, E, W)$, where N and E represent the node set and the edge set in G respectively. W is a weight matrix in which each element w_{ij} (where $i, j \in \{1, 2, \ldots, |N|\}$) denotes the weight of edge between node i and node j.

In a graph G, a node represents an individual in the targeted social network and an edge represents the social relationship (e.g. trust, friendship or collaboration) between two linked individuals. And the weight of an edge shows the connection strength. The larger the weight is, the stronger the corresponding relationship is. An unweighted social network can be viewed as a special case of weighted networks and it has the same weight for all edges.

In spread models of influence maximization, a node is defined as *active*, if and only if its corresponding individual adopts a new product or an idea. Otherwise, it is defined as *inactive*.

2.1. Influence Maximization

Before giving the definition of influence maximization, we first define the influence spread in a target network G.

Influence spread. The influence spread of a node set S, is defined as the expected number of active nodes with the given seed set S, which can be denoted as $\sigma(S)$.

And the influence maximization is defined as a problem on how to select k nodes (i.e. the seed set S) from a network G so that the influence spread $\sigma(S)$ is maximal. The influence maximization can be formally defined as a constrained optimization problem.

Influence maximization. Given a graph $G = (N, E, W)$ and an integer $k <\mid N \mid$, select k nodes as an initial node set $S = \{s_1, s_2, ..., s_k \mid s_i \in N, i = 1, 2, ..., k\}$, so that the influence spread $\sigma(S)$ is maximum for a specific spread model [56], which can be formulated as:

$$\begin{cases} S = \arg\max \sigma(S) \\ subject\ to \quad \mid S \mid = k. \end{cases} \tag{1}$$

The spread models widely used in influence maximization include the independent cascade model [8, 20], the weighted cascade model [10, 31] and the linear threshold model [24, 48]. As described above, the computation of influence spread $\sigma(S)$ for a given set S is #P-hard [61, 62] and the influence maximization has been proved as a NP-hard optimization problem [30, 39].

2.2. Diffusion Models

There are plenty of models that can be used to simulate the diffusion process in influence maximization. In this paper, two commonly used spread models called the independent cascade model and the weighted cascade model are chosen as our transmission mechanisms.

Both the IC and WC models belong to cascade models. The IC model was first introduced by Goldenberg et al. [20] and the WC model was developed by Kempe et al. [30] based on the IC model. In these two models, time unfolds in discrete steps. Each edge is assigned a probability value which is also called active probability or spread speed. Edge $(i, j) \in E$ with a probability value p_{ij} means that when the node i becomes active in step t, its inactive neighbor node j will be influenced with probability p_{ij} in the next step $(t+1)$.

The biggest difference between the two diffusion models lies in how to determine the active probability of each edge. In the IC model, the active probability p_{ij} with which node i activates its neighbor j can be computed as follows:

$$p_{ij} = 1 - (1 - p)^{w_{ij}} \tag{2}$$

where $p \in [0, 1]$ is the active probability and w_{ij} denotes the weight of the edge (i, j). Specifically, if the targeted social network is unweighted, the values of active probability for all edges equal to p [60].

The case for the WC model is different. The active probability between nodes i and j can be determined by the reciprocal of the in-degree number of node j, and it can be computed by Eq. (3).

$$p_{ij} = \frac{a_{ij}}{d_j^{(in)}} \tag{3}$$

where $a_{ij} = 1$ if the node j is one of neighbors of node i; and 0 otherwise. $d_j^{(in)}$ is the number of edges that point to node j in a directed graph G. Especially, if graph G is undirected, the $d_j^{(in)}$ in Eq. (3) is the degree of node j.

The spread process for cascade models. Given a social network $G = (N, E, W)$ and a seed set $S = \{s_1, s_2, ..., s_k\}$. At the initial time, only the nodes in the seed set S are active. Then, in the next step $(t+1)$, each active node in S_t has a single chance to influence its inactive neighbors with the corresponding active probability independently. If a neighboring node turns to be active, it will be added to the active set S_{t+1} and it obtains an opportunity to influence its inactive neighbors in the next time step. Once no new node becomes active in a time step, the spread process will be terminated and the total number of active set $|S + S_1 + ... + S_\xi|$ (where ξ denotes the terminal time) will be output as the final influence spread of the initial set S.

2.3. Related Works on Influence Maximization

In order to solve the influence maximization problem, two challenges should be settled: i) how to evaluate nodes' importance for the influence spread and ii) how to select influential nodes from the network. In previous literatures, many works have been recorded for solving the two challenges.

The degree centrality based method evaluates the influence spread of networks based on their degree [30]. A node with many neighboring nodes is considered to be influential in the spread process. Thus, degree method selects k nodes with the largest degree as the initial seed set for the influence maximization problem. Let d_i represents the degree of node i and it can be computed as follows:

$$d_i = \sum_{i \neq j} a_{ij} \tag{4}$$

As described in Eq. (3), $a_{ij} = 1$ represents that nodes i and j are linked; and 0 otherwise.

The closeness centrality based approach [10, 30, 31], also called the distance based method, evaluates the influence ability of a node according to the average shortest path length to the other nodes. Just like degree centrality method, the distance based method chooses seed nodes based on their average shortest length in the network. The average shortest path length l_i of node i can be formulated by Eq. (5).

$$l_i = \frac{1}{|N| - 1} \sum_{i \neq j} l_{ij} \tag{5}$$

where l_{ij} is the length of the shortest path between node i and node j.

Due to the great success of the Internet, PageRank [4] has been widely used in various areas. The PageRank technique can also be applied to evaluate the importance of a node in a network. In PageRank, the PageRank score of node i can be obtained by Eq. (6).

$$I_{(pr),i} = \frac{1 - \varepsilon}{|N|} + \varepsilon \sum_{j \in N} \frac{a_{ij} I_{(pr),j}}{d_j^{(out)}} \tag{6}$$

where ε is a damping factor and $d_j^{(out)}$ is defined as the out-degree of node j.

In order to improve the efficiency of basic greedy algorithm, Jiang et al. [29] introduced a novel metric named expected diffusion value (EDV for short) in the IC model to replace the expected diffusion simulation. The EDV metric approximates $\sigma(S)$ by computing how many neighbors of the selected node set S are expected to be influenced, which can be computed as follows:

$$EDV(S) = k + \sum_{i \in N_S^{(1)} \setminus S} (1 - (1 - p)^{\tau(i)}) \tag{7}$$

where $N_S^{(1)}$ represents the direct neighbors (i.e. one-hop area) of S, and $i \in N_S^{(1)} \setminus S$ represents that i belongs to $N_S^{(1)}$ but not to S. p is the preset value for activation probability in the IC model and $\tau(i)$ denotes the number of links between node i and the initial node set S.

Based on EDV, a simulated annealing method is proposed to find the top-k influential nodes with the maximum value of EDV, which is named SAEDV. Experiments show that SAEDV can obtain a node set with a high quality after millions iterations [29].

2.4. Particle Swarm Optimization

The main idea of particle swarm optimization (PSO) originates from the movement of bird flocks. PSO can find out the optimal solution in the search space just like bird flocks searching for corns. In PSO, each particle has a position and a velocity vectors. The position vector denotes the current solution of the corresponding particle and the velocity vector is utilized to provide the direction and adjust the particle's position to the optimal solution [2].

PSO was originally proposed to solve continuous optimization. Kennedy and Eberhart first introduced PSO into discrete optimization problems and proposed a discrete binary version of the PSO, called BPSO [33]. In recent years, many of PSO have been proposed for various applications. Among them, the version introduced by Shi and Eberhart

[51] has been widely used. Let $\mathbf{X_i} = (x_{i1}, x_{i2}, ..., x_{ik})$ and $\mathbf{V_i} = (v_{i1}, v_{i2}, ..., v_{ik})$ separately represents the position vector and velocity vector of particle i ($i = 1, 2, ..., n$), where k is the size of the seed set and n represents the size of population in PSO. According to the study by Shi and Eberhart [17], the update rules for velocity and position vectors can be formulated as follows:

$$\mathbf{V_i} \leftarrow \omega \mathbf{V_i} + c_1 r_1(\mathbf{Pbest_i} - \mathbf{X_i}) + c_2 r_2(\mathbf{Gbest} - \mathbf{X_i}) \tag{8}$$

$$\mathbf{X_i} \leftarrow \mathbf{X_i} + \mathbf{V_i} \tag{9}$$

where ω in Eq. (8) is called the inertia weight, the $\mathbf{Pbest_i} = (pbest_{i1}, pbest_{i2}, ..., pbest_{ik})$ is defined as the personal best position of particle i (i.e. the local optimal position vector for particle i), and $\mathbf{Gbest} = (gbest_1, gbest_2, ..., gbest_k)$ represents the global best position in the swarm. Parameters c_1 and c_2 are called learn factors, and r_1 and $r_2 \in [0, 1]$ denote random numbers.

3. Model and Algorithm

In this section, we first give a description for the proposed optimization model. Then the detailed implementations of the proposed DPSO algorithm and the analysis of the computational complexity are described.

3.1. Objective function

The computation of the influence spread $\sigma(S)$ and the discovery of a seed set with the best influence spread in a social network are two intractable problems in influence maximization, which have been proved to be #P-hard and NP-hard, respectively.

In order to compute the influence spread, traditional Monte Carlo simulation needs to run at least tens of thousands times to obtain an accurate estimation, which is very time-consuming. To tackle this problem, Jiang et al. proposed a fast EDV function to replace the expensive diffusion simulations in the IC model [29]. However, the EDV only takes the neighbors of seed set S into consideration, which can only effectively tackle the situation with a small active probability in the IC model. Hence, it is essential to design a function that considers a good trade-off between running time and accuracy to compute the influence spread of a node set.

Actually, the influence spread of an individual's opinion in social networks is limited within the circle of friends' friends' friends (i.e. three-hop area), because there exists an intrinsic-decay, which is called the Three Degree Theory [12]. In the recent works, Pei et al. further pointed that the global influence of a node in the network depends on the influence in the two-hop area [46]. Following those studies, we develop a local influence estimation to approximate the influence spread within the two-hop area (the neighbors' neighbors) for a node set S. Moreover, we introduce the LIE function to compute the influence spread in both the IC and WC models.

The expected influence spread of one-hop area for a node set S can be computed as:

$$\sigma_1(S) = \sigma_0(S) + \sigma_1^*(S) = k + \sum_{i \in N_S^{(1)} \setminus S} (1 - \prod_{(i,j) \in E, j \in S} (1 - p_{ij})) \tag{10}$$

where p_{ij} represents the activation probability of node i activating node j.

Based on $\sigma_1(S)$, the extended LIE function can be formulated as follows:

$$\begin{aligned} LIE(S) &= \sigma_0(S) + \sigma_1^*(S) + \widetilde{\sigma}_2(S) \\ &= k + \sigma_1^*(S) + \frac{\sigma_1^*(S)}{|N_S^{(1)} \setminus S|} \sum_{u \in N_S^{(2)} \setminus S} p_u^* d_u^* \\ &= k + (1 + \frac{1}{|N_S^{(1)} \setminus S|} \sum_{u \in N_S^{(2)} \setminus S} p_u^* d_u^*) \sigma_1^*(S) \\ &= k + (1 + \frac{1}{|N_S^{(1)} \setminus S|} \sum_{u \in N_S^{(2)} \setminus S} p_u^* d_u^*) \sum_{i \in N_S^{(1)} \setminus S} (1 - \prod_{(i,j) \in E, j \in S} (1 - p_{ij}))) \end{aligned} \tag{11}$$

where $N_S^{(1)}$ and $N_S^{(2)}$ represents the S's one-hop and two-hop area, respectively. The parameter p_u^* is the constant active probability of node i, and it corresponds to p and $1/d_i^{(in)}$ in the IC model and the WC model respectively. d_u^* is the number of edges of node u within $N_S^{(1)}$ and $N_S^{(2)}$, which represents the number of activated probability for node u. Generally, $d_u^* \leq d_u$.

As shown in Eq. (11), the computational complexity of LIE function is $O(k\bar{D}^2)$, where $\bar{D}$ is the average degree for the given social network. The LIE function can be utilized to estimate the two-hop influence spread in both the IC and WC models.

Based on this work, the problem of influence maximization is converted to be the optimization of an extended LIE function and the optimal seed set can be obtained by maximizing the LIE value.

3.2. Proposed algorithm

The whole framework of the proposed DPSO algorithm for influence maximization is given in **Algorithm 1**.

Algorithm 1 Framework of DPSO algorithm for influence maximization

1: **Input**: Graph $G = (N, E, W)$, the size of particle swarm n, the number of iterations g_{max}, the inertia weight ω, the learn factors c_1 and c_2 and the size of the seed set k.
2: **Step1** Initialization:
3: **Step1.1** Initialize position vector: $\mathbf{X} \leftarrow$ Degree_Based_Initialization(G,n,k);
4: **Step1.2** Initialize **Pbest** vector: **Pbest** $\leftarrow$ Degree_Based_Initialization(G,n,k);
5: **Step1.3** Initialize velocity vector: $\mathbf{V} \leftarrow \mathbf{0}$;
6: **Step2** According to the fitness value, select out the global best position vector **Gbest**;
7: **Step3** Begin cycling and set $g \leftarrow 0$;
8: **Step3.1** Update the velocity vector **V**;
9: **Step3.2** Update the position vector **X**;
10: **Step3.3** Update the **Pbest** and select out the best particle **Gbest*** in the current iteration;
11: **Step3.4** Employ the local search operation on **Gbest***: **Gbest**$'$ $\leftarrow$ Local_Search(**Gbest***);
12: **Step3.5** Update the **Gbest**: **Gbest** $\leftarrow$ Max(**Gbest**$'$, **Gbest**)
13: **Step4** Stop criteria: if $g = g_{max}$, stop the algorithm; otherwise, let $g \leftarrow g + 1$ and go to **Step3.1**;
14: **Output**: Output the best position **Gbest** as the seed set S.

The function Max(**Gbest**$'$,**Gbest**) in **Algorithm 1** can select the particle with larger LIE value. The detailed descriptions of particles' representation, initialization (Degree_Based_Initialization($\cdot$)), update rules and network-based local search operator (Local_Search($\cdot$)) in **Algorithm 1** are given in the following.

3.2.1. Representation

In the proposed DPSO algorithm, both position and velocity vectors of particles are redefined in a discrete form to make the DPSO applicable for the influence maximization problem.

Definition 1 *(Position vector).* In DPSO, the position for particle i is encoded as a k-dimensional integer vector $\mathbf{X_i} = (x_{i1}, x_{i2}, ..., x_{ik})$ (where $i = 1, 2, ..., n$) and denotes a candidate seed set for influence maximization in the targeted network. Each element x_{ij} (where $x_{ij} \leq \mid N \mid$ and $j = 1, 2, ..., k$) in $\mathbf{X_i}$ is the sequence number of nodes. For example, if the size of seed set k=5 and the position vector for particle i is $\mathbf{X_i} = (1, 2, 8, 10, 12)$, it means that nodes 1, 2, 8, 10 and 12 are selected as a seed set in the targeted social network.

At the beginning, position vectors are initialized by a degree based heuristic method and the details are given in subsection 3.2.2.

Definition 2 *(Velocity vector).* The velocity of particle i is defined as $\mathbf{V_i} = (v_{i1}, v_{i2}, ..., v_{ik})$, which is the judgment for the current position. In the iteration, each particle learns from the leader and its own prior knowledge, and then estimates how to change directions towards a more promising position. For a velocity vector $\mathbf{V_i}$, each $v_{ij} \in \{0, 1\}$. If $v_{ij} = 1$, the corresponding value needs to be changed; and 0 otherwise. At the initial time, **V** is set to **0**.

3.2.2. Initialization

In order to speed up the convergence of the proposed DPSO algorithm, a degree based heuristic method is adopted to perform initialization for particles' position vectors. The main procedure is outlined in **Algorithm 2**.

Algorithm 2 Degree_Based_Initialization(G,n,k)

```
1: Input: Graph G = (N, E, W), swarm size n, the size of seed set k.
2: for each i ≤ n do
3:     X_i ←Degree(G, k);
4:     for each element x_ij ∈ X_i do
5:         if random>0.5 then
6:             x_ij ←Replace(x_ij, N);
7:         end if
8:     end for
9: end for
10: Output: The initial position vector X.
```

As shown in **Algorithm 2**, all particles are initialized by the degree based method which selects k influential nodes with the highest degree in graph G and the procedure is denoted as Degree(G, k) in **Algorithm 2**. However, the degree heuristic method is easy to generate a swarm lacking of diversity, because all of the particles have the same starting position. Here, a turbulence operator is adopted to promote the diversity randomly. For each element of position vectors, a probability in the interval of [0, 1] is generated. If the generated value is larger than 0.5, the corresponding value x_{ij} will be randomly replaced with another new node that is different from the others in $\mathbf{X_i}$; otherwise, x_{ij} stays unchanged. The function Replace(x_{ij}, N) replaces the element x_{ij} with another node selected from the node set N randomly, and guarantees that there is no repeated node in the particle $\mathbf{X_i}$ after the replacement.

The degree based initialization method described in **Algorithm 2** can be completed easily and quickly. After the initialization, a swarm with good performance in both influence spread and diversity can be obtained.

3.2.3. Update Rules

To solve the influence maximization problem effectively, the update rules for both position and velocity are also redesigned in a discrete form.

(1) Update rule for velocity

In PSO, the particle velocity vector is utilized to provide a reliable guidance for particles to search the promising area, thus it plays a significant role in the whole process of optimization [6]. In the proposed DPSO algorithm, the update rule of the velocity vector is redefined as follows:

$$V_i \leftarrow \mathbf{H}(\omega \mathbf{V_i} + c_1 r_1(\mathbf{Pbest_i} \cap \mathbf{X_i}) + c_2 r_2(\mathbf{Gbest} \cap \mathbf{X_i})) \tag{12}$$

Eq. (12) has the similar form as that of BPSO [33]. But its key components are different from those in BPSO.

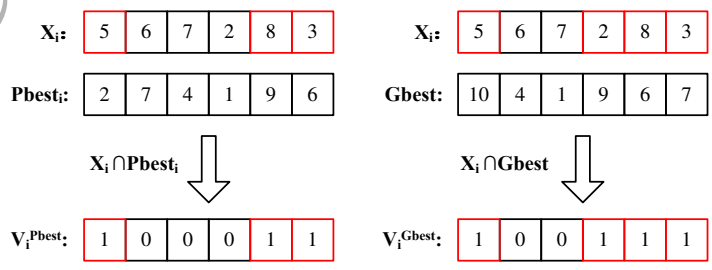


Fig. 1: An illustration of operator "∩". In mathematical terms, $\{5, 6, 7, 2, 8, 3\} \cap \{2, 7, 4, 1, 9, 6\} = \{6, 7, 2\}$. Thus, the corresponding elements of 6,7 and 2 in $\mathbf{X_i}$ are set to 0 and the others are 1 in $\mathbf{V_i^{Pbest}}$, namely $\mathbf{V_i^{Pbest}} = (1, 0, 0, 0, 1, 1)$. Similarly, $\mathbf{V_i^{Gbest}} = (1, 0, 0, 1, 1, 1)$.

The operator "$\cap$" in Eq. (12) is defined as a similar intersection operation. A detailed illustration for this operator is shown in Fig. 1. Assuming that the number of the initial set $k = 6$, the position vector of particle i is $\mathbf{X_i} = (5, 6, 7, 2, 8, 3)$. $\mathbf{Pbest_i} = (2, 7, 4, 1, 9, 6)$ represents the personal best position of particle i and $\mathbf{Gbest} = (10, 4, 1, 9, 6, 7)$ is the global best position in the swarm. Firstly, intersect vector $\mathbf{X_i}$ and $\mathbf{Pbest_i}$ mathematically and the same elements in both two vectors can be obtained. They are 6, 7 and 2 respectively as shown in Fig. 1. Then the corresponding elements in $\mathbf{V_i^{Pbest}}$ are set to 0, which denotes that those directions may be potential and tend to be reserved. Conversely, the other elements in $\mathbf{V_i^{Pbest}}$ are set to 1, which indicates that those directions are not good choices and they need to be adjusted. Thus, the $\mathbf{V_i^{Pbest}}$ can be computed as $(1, 0, 0, 0, 1, 1)$. In a similar way, the $\mathbf{V_i^{Gbest}}$ can be also computed as $(1, 0, 0, 1, 1, 1)$.

Once $\mathbf{V_i^{Pbest}}$ and $\mathbf{V_i^{Gbest}}$ are obtained, the new velocity vector of the particle i can be calculated easily according to Eq. (12). The argument of $\mathbf{H}(\cdot)$ is a position vector. Assuming that the argument is $\mathbf{X_i}$, the function $\mathbf{H}(\mathbf{X_i})$ can be represented as $\mathbf{H}(\mathbf{X_i}) = (h_1(x_{i1}), h_2(x_{i2}), ..., h_k(x_{ik}))$ and $h_j(x_{ij})$ (where $j = 1, 2, ..., k$) is defined as a threshold function shown as follows.

$$h_j(x_{ij}) = \begin{cases} 0, & if\ x_{ij} < 2; \\ 1, & if\ x_{ij} \geq 2; \end{cases} \tag{13}$$

For example, given coefficients $c_1 r_1 = 0.9$ and $c_2 r_2 = 1.3$, then the velocity vector $\mathbf{V_i}$ shown in Fig. 1 can be calculated as $\mathbf{V_i} = \mathbf{H}(0.9 * (1, 0, 0, 0, 1, 1) + 1.3 * (1, 0, 0, 1, 1, 1)) = \mathbf{H}((2.2, 0, 0, 1.3, 2.2, 2.2)) = (1, 0, 0, 0, 1, 1)$.

(2) Update rule for position

The redefined the update rule can be described as follows:

$$\mathbf{X_i} \leftarrow \mathbf{X_i} \oplus \mathbf{V_i} \tag{14}$$

The operator "$\oplus$" plays an important role in driving the particles to the promising regions. Assuming that $\mathbf{X_i} \oplus \mathbf{V_i} = \mathbf{X_i'} = (x_{i1}', x_{i2}', ..., x_{ik}')$, elements in the new position $\mathbf{X_i'}$ can be updated by Eq. (15).

$$x_{ij}' = \begin{cases} x_{ij}, & if\ v_{ij} = 0; \\ Replace(x_{ij}, N), & if\ v_{ij} = 1; \end{cases} \tag{15}$$

As stated in subsection 2.1, N is the node set of the targeted network G. As the same in Algorithm 2, Replace(x_{ij}, N) in Eq. (15) is a function that can replace the element x_{ij} with a random node in the node set N and it can guarantee that there is no repeated node in $\mathbf{X_i}$ after the replacement at the same time.

As described above, the update rules for both velocity and position are simple but effective. In each iteration, the velocity updating can provide a reliable direction guidance for the whole swarm. Then, the particles adjust their directions according to the update rule to fly in the promising directions.

3.2.4. Local Search

In our DPSO algorithm, particles can find the promising regions by mutual learning and the leader's guidance during each iteration. However, it is slow for the whole swarm to explore blindly. A network-specific local search strategy is proposed to improve the convergence speed. It works on the leader particle and its pseudocode is described in **Algorithm 3**.

$N_{x_{bi}}^{(1)}$ in line 5 represents the neighbor set of node x_{bi}. As shown in **Algorithm 3**, our local search is designed with a greedy strategy and it can guide the leader particle to the promising solutions by using the information of neighbors.

In the proposed DPSO algorithm, the population-based PSO can find multiple promising solutions and the greedy local search strategy based on neighborhood information can find the optimal solution around the promising regions. Combining the PSO and the designed local search operator, the proposed DPSO method can achieve better performances in both exploration and exploitation [47, 49].

3.3. Computational Complexity Analysis

Here, we analyze the computational complexity of the proposed DPSO algorithm. As introduced in subsection 3.2.2, the degree based initialization method needs $O(n \cdot k)$ basic operations. For the suggested DPSO, the main time complexity lies in **Step 3** in **Algorithm 1**. Here, we use n and g_{max} to denote the size of particle swarm and the number

Algorithm 3 Local_Search($\mathbf{X_a}$)

Input: Particle $\mathbf{X_a}$.
$\mathbf{X_b} \leftarrow \mathbf{X_a}$;
for each element $x_{bi} \in \mathbf{X_b}$ **do**
    Flag←FALSE;
    $Neighbors \leftarrow N^{(1)}_{x_{bi}}$
    repeat
        x_{bi} ←Replace(x_{bi},$Neighbors$);
        if LIE($\mathbf{X_b}$)>LIE($\mathbf{X_a}$) **then**
            $\mathbf{X_a} \leftarrow \mathbf{X_b}$;
        else
            Flag←TRUE;
        end if
    until Flag is TRUE
    $\mathbf{X_b} \leftarrow \mathbf{X_a}$;
end for
Output: Particle $\mathbf{X_b}$.

of iterations. **Step 3.1** needs $O(k \cdot \log k \cdot n)$ basic operations. **Step 3.2** and **Step 3.3** need $O(k \cdot n)$ basic operations. **Step 3.4** requires $O(k \cdot \bar{D})$ basic operations, where $\bar{D}$ is the averaged degree of a network. **Step 3.5** requires $O(1)$ basic operation. Thus, the worst case time complexity is $O(k \cdot \log k \cdot n) + 4O(k \cdot n) + O(k \cdot \bar{D}) + O(1)$. Besides, the time complexity of the object function $LIE(\cdot)$ is $O(k \cdot \bar{D}^2)$. According to the rules of the symbol O, in the worst case, the time complexity of the proposed DPSO is $O(k^2 \cdot \log k \cdot n \cdot \bar{D}^2 \cdot g_{max})$, where n and g_{max} are both set to 100 in our experiments.

4. Experimental Results

In this section, the experiment for parameters are first given. Then to test the effectiveness of the designed local search strategy, we compare DPSO with the variant version without the local search, called DPSO-nls. Finally, the experimental results of different algorithms are compared in terms of the influence spread and the running time. Note that, when finding the initial node set, each algorithm uses its own objective functions. More specifically, DPSO and DPSO-nls use the proposed LIE, SAEDV uses the EDV function and CELF++ uses the influence spread obtained by the Monte Carlo simulation as the optimization objective. In classical influence maximization algorithms, e.g., CELF++ [23], SAEDV [29], etc., the Monte Carlo spread simulation method are used to evaluate the performance of different algorithms. When comparing the experimental results, all the influence spread are calculated by Monte Carlo simulation method with 10000 runs [30, 31].

4.1. Comparison Algorithms and Experimental Networks

Four influence maximization algorithms available in the literature, including the state-of-the-art CELF++ greedy method [37], the degree centrality method, PageRank [4] and SAEDV [29], are selected to compare with our algorithm. Moreover, a variant version of the suggested DPSO algorithm without local search operator, called DPSO-nls, is also compared with DPSO algorithm. In our experiments, the degree centrality method is denoted as Degree.

Among the four experimental networks, NetInfective is a human contact network, which describes the face-to-face behaviors of people during the exhibition INFECTIOUS. The rest three networks, including NetScience, NetGRQC and NetHEPT, are co-author networks and describe the cooperative relationships among the scientists in different domains. In the NetInfective and NetHEPT networks, multiple edges may exist between two nodes. The NetScience network is a weighted network and the others are unweighted. In our experiments, the active probability p of the IC model is set to 0.01 in the NetInfective, NetGRQC and NetHEPT networks. Due to the sparsity and small scale, p is set to 0.05 in the NetScience network. The statistic characters of the four experimental social networks are listed in Table 1.

Table 1: Statistic characters of the four real-world social networks.

Networks	$\|N\|$	$\|E\|$	$\bar{D}$	Reference
NetInfective	410	17,298	84.38	[28]
NetScience	1,589	2,742	3.45	[41]
NetGRQC	5,242	14,495	5.53	[36]
NetHEPT	15,233	58,891	7.73	[10]

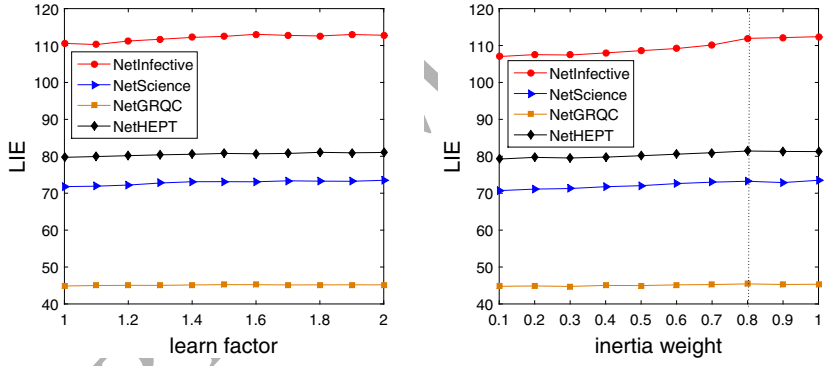
The proposed algorithm and all comparison algorithms are written in MATLAB language and each independently run 30 times on a PC with Intel (R) Core (TM), 2 CPU, 2.33GHz, 4G memory.

4.2. Experiments for Parameters

First, we conduct experiments to estimate the influence of parameters for the final LIE value in the proposed DPSO algorithm. We give an example of the DPSO algorithm in the IC model. The size of initial seed set k is set to 30 and all of experiments are conducted for 30 times independently.

4.2.1. Learn factors and inertia weight

To verify the values of learn factors and the inertia weight, the number of iterations g_{max} and the size of the swarm n are both set to 100. First, we keep the inertia weight $w = 0.8$ and vary the values of learn factors c_1 and c_2 between [1, 2] with interval 0.1. As for the inertia weight, the values of c_1 and c_2 are fixed as 2 and the parameter w changes in [0.1, 1]. Fig. 2 shows the variations of the LIE value with different learn factors and the inertia weight.



(a) Learn factors c_1 and c_2. (b) Inertia weight w.

Fig. 2: Variations of the fitness value with different learn factors and inertia weight.

As shown in Fig. 2(a), with the increase of c_1 and c_2, the fitness value increases in all the four real-world social networks. Therefore, c_1 and c_2 in our DPSO algorithm are set to 2. For the parameter w, the variation curves of the NetInfective and NetHEPT networks increase with the inertia weight w varying from 0.1 to 1.0 and the performances of DPSO with w=0.8, 0.9 and 1.0 are approximate. In the other two networks, NetScience and NetGRQC, the proposed DPSO algorithm performs the best with w=0.8. Thus, we set inertia weight to 0.8 on the four real-world social networks.

4.2.2. The number of iterations and the size of swarm

To analyze the performance of DPSO with different values of n and g_{max}, we set c_1 and c_2 to 2, and w to 0.8. Fig. 3 and Fig. 4 show the performance of DPSO with different values of n and g_{max}, respectively.

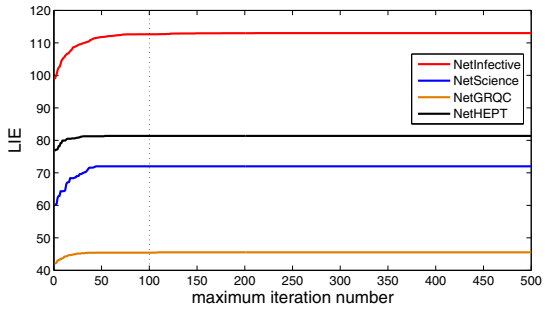
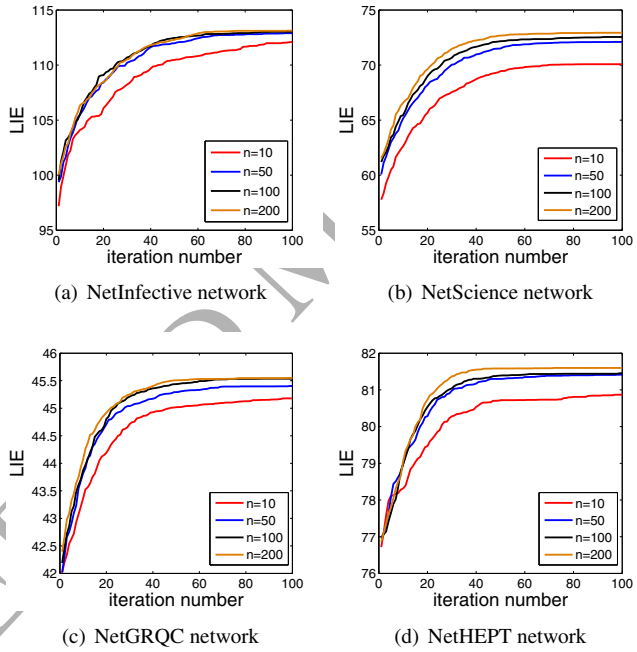


Fig. 3: Variations of fitness value within 500 iterations in the four networks.

Fig. 3 shows the variation curves of fitness value within 500 iterations in the four real-world social networks. As shown in Fig. 3, DPSO has achieved convergence within 100 iterations for the four networks. Here, DPSO with the maximum iteration number g_{max}=100 is a reasonable choice when we consider a tradeoff between the performance and the computational time.



(a) NetInfective network (b) NetScience network

(c) NetGRQC network (d) NetHEPT network

Fig. 4: Variations of fitness value with different parameter n.

As shown in Fig 4, the final fitness values within 100 iterations increase with the rising parameter n, which denotes that the larger the size of the particle swarm is, the better the performance of DPSO has. But the fitness values with n=100 and n=200 are very approximate. Considering the tradeoff between the performance and efficiency, we set the parameter n to 100.

According to the analysis above, the learn factors c_1 and c_2, the inertia weight w, the maximum iteration number g_{max} and the size of particle swarm n in the proposed DPSO are set to 2, 0.8, 100 and 100, respectively. Moreover, the experiments for the parameters of other algorithms are also conducted. Finally, the dumped factor ε for PageRank is set to 0.15; the paramters for the SAEDV are set as follows: the initial temperature T_0=500,000, the final temperature T_f=8,000, the number of the inner loop q=300 and the temperature drop after each inner loop $\triangle T$ = 2000. The

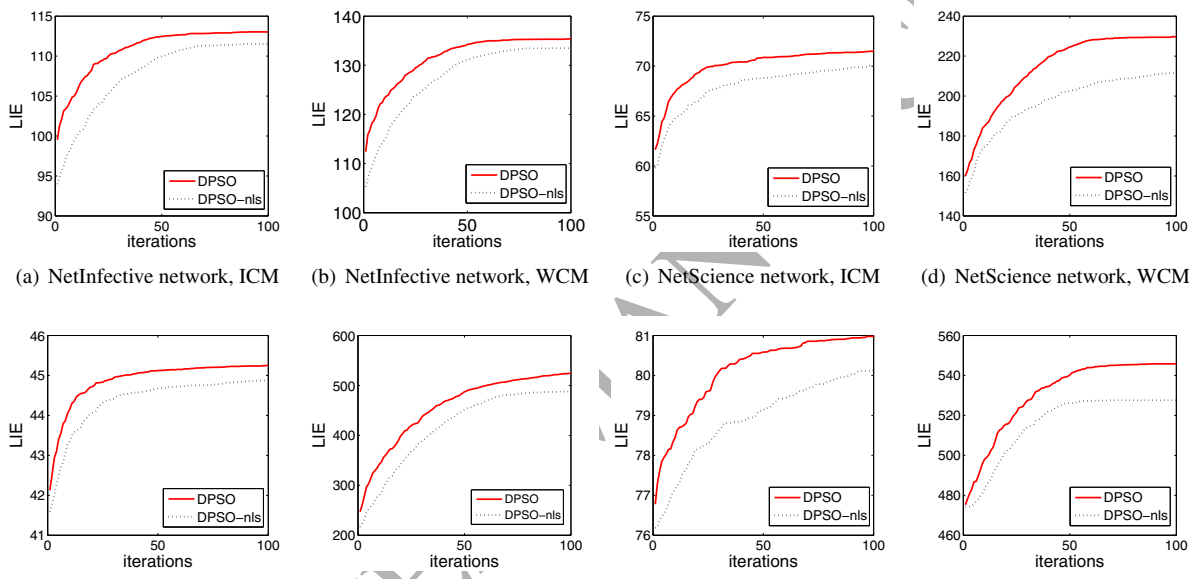
parameters tuning for DPSO and SAEDV may be not the best, but it guarantees that DPSO and SAEDV can achieve convergence on the four networks while considering a tradeoff between the performance and the computational time.

4.3. Comparison Experiments in the Real-world Social Networks

In this part, extensive experiments on the four real-world social networks are carried to validate the effectiveness and the efficiency of the proposed DPSO. In our experiments, the number of seed set k varies from 3 to 30 and the number of Monte Carlo simulations for the selected seed sets is set to 10000 in all cases.

4.3.1. Experiments for the Designed Local Search

In order to exhibit the influence of the designed local search operator, we compare the speed of convergence between DPSO and its variant version DPSO-nls. Fig. 5 displays the values of the LIE obtained by DPSO and DPSO-nls on the four real-world networks with the same parameters. The size of the initial set k is set to 30.



(a) NetInfective network, ICM (b) NetInfective network, WCM (c) NetScience network, ICM (d) NetScience network, WCM

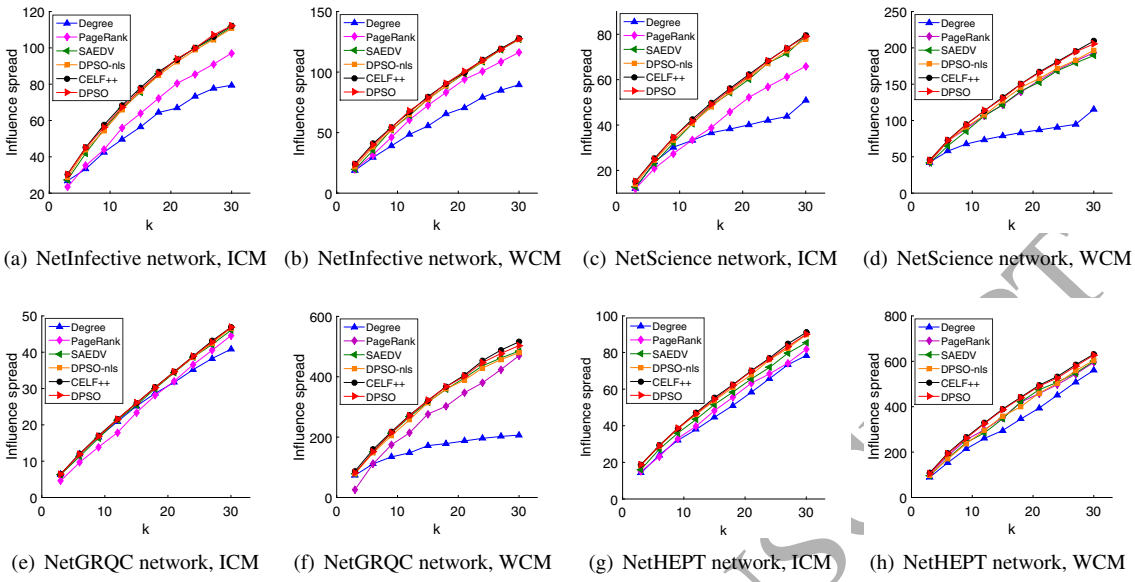
(e) NetGRQC network, ICM (f) NetGRQC network, WCM (g) NetHEPT network, ICM (h) NetHEPT network, WCM

Fig. 5: Comparisons of convergence between DPSO and DPSO-nls. The number of iteration and the size of the swarm are both set to 100, parameters $c_1, c_2 = 2$, $w = 0.8$ and the size of seed set k is set to 30.

As shown in Fig. 5, DPSO can find the maximum value of LIE within 50 iterations, while DPSO-nls cannot find the optimal seed set during 100 iterations. Especially in the WC model, the search performance of DPSO-nls is poor and it can hardly find a candidate seed set with good quality in influence spread within 100 iterations. However, DPSO outperforms DPSO-nls with large margins and it has stabilized after 50 iterations on the four real-world social networks. From those comparisons, it can be concluded that the designed local search operator plays an important role in the search process. The neighbor based local search improves the searching performance and speeds up the convergence of DPSO at the same time.

4.3.2. Comparisons for the influence spread

The influence spread results of different algorithms in both the IC and WC models are shown in Fig. 6.



(a) NetInfective network, ICM (b) NetInfective network, WCM (c) NetScience network, ICM (d) NetScience network, WCM

(e) NetGRQC network, ICM (f) NetGRQC network, WCM (g) NetHEPT network, ICM (h) NetHEPT network, WCM

Fig. 6: Influence spread on the four real-world social networks in both the IC and WC models. In the IC model, the active probability p=0.05 for NetScience and p = 0.01 for the other networks.

As shown in Figs. 6(a) and 6(b), DPSO, DPSO-nls, CELF++ and SAEDV have similar performance on the NetInfective and all of them exceed Degree and PageRank in influence spread. Furthermore, DPSO and CELF++ outperform DPSO-nls and SAEDV a little bit in both the IC model and the WC model. Moreover, PageRank performs better than Degree.

In the NetScience network, the suggested DPSO and the state-of-the-art CELF++ both have the best performance, whether for the IC model or the WC model. DPSO-nls and SAEDV can match CELF++ and DPSO in the IC model, but they both yield to CELF++ and DPSO in the WC model. As shown in Fig. 6(d), the SAEDV and PageRank have a close performance, which perform worse than DPSO-nls. Focusing on DPSO-nls, we can find that the search ability of DPSO-nls becomes worse with the increase of k in Fig. 6(d), because the search space gets huger with the increase of the size of seed set. As a comparison, DPSO with local search strategy has the more stable performance on the search. PageRank performs better in the WC model than in the IC model.

Figs. 6(e) and 6(f) show the influence spread in the NetGRQC network. The results of SAEDV, DPSO and CELF++ can achieve the best influence spread in the IC model. DPSO-nls outperforms SAEDV and Degree. When $k < 21$, Degree exceeds PageRank. While the performance of Degree is inferior to the other algorithms in the case of $k > 21$. For the WC model, Degree has poor performance and the influence spread of the other four algorithms are much more than that of Degree, which shows that Degree has instable performance in the influence spread. As shown in Fig. 6(f), DPSO has a competitive performance with CELF++ when k is less than 21 and it is inferior to CELF++ with larger settings of k. The SAEDV performs better than DPSO-nls and they both perform worse than DPSO and CELF++.

As for the NetHEPT network, the proposed DPSO algorithm has a competitive performance compared with CELF++ algorithm and outperforms the other algorithms for the influence spread results in both the IC and WC models. Besides, the SAEDV algorithm outperforms DPSO-nls, PageRank and Degree in Figs. 6(g) and 6(h). We can see that PageRank only performs better than Degree in the IC model.

4.3.3. Statistical test for the influence spread

In this part, the statistical test is carried to detect the significant differences in the influence spread results. First, we apply the Friedman test [18] and the Iman-Davenport test [27] to see whether there are significant differences in the results. If there are statistical differences, the Holm-Bonferroni procedure [14, 26] is used as the post hoc procedure. The Holm-Bonferroni procedure uses the best algorithm as control algorithm, and detects the concrete differences

between the control algorithm and the comparison algorithms. We set the level of confidence α to 0.05 in all cases. Detailed descriptions of the tests are presented in [14, 19].

Both the Friedman test and the Iman-Davenport test indicate that there are significant differences in the influence spread results of the IC model for each k value. Therefore, the Holm-Bonferroni procedure is applied as the post hoc procedure to detect the concrete differences between the algorithms.

The Holm-Bonferroni procedure [14]: Given the statistical values achieved by N_a algorithms on the N_p problems. The algorithm showing the best performance is chosen as the control method. Let $p_1, p_2, ..., p_{N_a-1}$ be the ordered p-values (smallest to largest) for the reminding algorithms, i.e., $p_1 < p_2 < ... < p_{N_a-1}$, and let $H_1, H_2, ..., H_{N_a-1}$ be the corresponding hypotheses. The Holm-Bonferroni procedure rejects H_1 to H_{i-1} $(1 < i < N_a - 1)$ if i is the smallest integer meeting $p_i > \alpha/(N_a - i)$. More specifically, the procedure starts with the most significant p-values. If $p_1 < \alpha/(N_a - 1)$, the corresponding hypothesis H_1 is rejected and compare p_2 with $\alpha/(N_a - 2)$. If H_2 is rejected, the test proceeds until a null hypothesis cannot be rejected, and then all the remaining hypotheses are retained.

The p-value/i results based on the Holm-Bonferroni procedure for the maximal and average influence spread results (IC model) are shown in Table 2 and Table 3, respectively. In the tables, "—" means the corresponding algorithm is chosen as the control method. The highlighted value means the corresponding hypothesis is rejected.

Table 2: The p-value/i results based on the Holm-Bonferroni procedure on the maximal influence spread results (IC model).

Algorithms	$k = 3$	$k = 6$	$k = 9$	$k = 12$	$k = 15$	$k = 18$	$k = 21$	$k = 24$	$k = 27$	$k = 30$
DPSO	—	—	—	—	—	—	—	—	—	—
CELF++	0.4386/5	0.5137/5	0.7054/5	1.0/5	0.5707/5	0.5706/5	0.3718/5	0.5708/5	0.4497/5	0.4497/4
DPSO-nls	0.4386/4	0.5137/4	0.6894/4	1.0/4	0.5136/4	0.3718/4	0.3718/4	0.5576/4	0.3917/3	0.3717/4
SAEDV	0.0545/3	0.04206/3	0.1763/3	0.2669/3	0.1129/3	0.1763/3	0.1129/3	0.5576/3	0.3917/4	0.2669/3
PageRank	**0.0023/1**	**0.0016/1**	0.0124/1	0.0326/2	**0.0099/2**	**0.0099/2**	**0.0099/2**	0.0326/2	0.0183/2	0.0183/2
Degree	0.0245/2	**0.0099/2**	0.0183/2	0.0124/1	**0.0033/1**	**0.0033/1**	**0.0008/1**	**0.0033/1**	**0.0016/1**	**0.0016/1**

Table 3: The p-value/i results based on the Holm-Bonferroni procedure on the average influence spread results (IC model).

Algorithms	$k = 3$	$k = 6$	$k = 9$	$k = 12$	$k = 15$	$k = 18$	$k = 21$	$k = 24$	$k = 27$	$k = 30$
DPSO	0.8501/5	0.3447/5	—	0.4496/5	0.8501/5	0.4496/5	0.8501/5	0.4497/5	0.7055/5	0.5707/5
CELF++	—	—	1.0/5	—	—	—	—	—	—	—
DPSO-nls	0.3717/4	0.1779/4	0.3717/4	0.1779/4	0.5137/4	0.1779/4	0.3917/4	0.1129/3	0.2669/3	0.2669/3
SAEDV	0.1129/3	0.0326/2	0.1763/3	0.1129/3	0.2669/3	0.1129/3	0.3917/3	0.1779/4	0.2669/4	0.2669/4
PageRank	**0.00657/1**	**0.0016/1**	0.0124/1	**0.0053/2**	0.0183/2	**0.0053/2**	0.0326/2	**0.0099/2**	0.0183/2	0.0183/2
Degree	0.0933/2	0.0421/3	0.0183/2	**0.0016/1**	**0.0065/1**	**0.0016/1**	**0.0033/1**	**0.0008/1**	**0.0016/1**	**0.0016/1**

The results based on the Holm-Bonferroni procedure shown in Table 2 demonstrate that the proposed DPSO outperforms the comparison algorithms in term of the maximal influence spread values, except that DPSO has approximate results with CELF++ and DPSO-nls when $k = 12$. From the results on the average values of the influence spread shown in Table 3, we can conclude that DPSO has shown comparable performance to CELF++ especially when $k = 3, 9, 15$, and 21. The two algorithms significantly outperform the other four algorithms.

Table 4: The p-value/i results based on the Holm-Bonferroni procedure on the maximal influence spread results (WC model).

Algorithms	$k = 3$	$k = 6$	$k = 9$	$k = 12$	$k = 15$	$k = 18$	$k = 21$	$k = 24$	$k = 27$	$k = 30$
DPSO	—	—	0.7055/5	—	—	—	—	0.7055/5	—	—
CELF++	0.8501/5	0.7055/5	—	0.5708/5	0.7055/5	0.3447/5	0.5708/5	—	0.5708/5	0.7055/5
DPSO-nls	0.6894/4	0.1779/4	0.2611/4	0.3718/4	0.2611/4	0.1176/4	0.2669/4	0.2611/4	0.2669/4	0.2669/4
SAEDV	0.0561/3	0.0934/3	0.0701/3	0.1129/3	0.1763/3	0.1129/3	0.2669/3	0.1129/3	0.2669/3	0.2669/3
PageRank	0.0561/2	0.0934/2	0.0561/2	0.0326/2	0.0183/2	0.0326/2	0.0183/2	0.0326/2	0.0183/2	0.0183/2
Degree	0.01248/1	**0.0016/1**	**0.0016/1**	**0.0016/1**	**0.0016/1**	**0.0008/1**	**0.0016/1**	**0.0016/1**	**0.0016/1**	**0.0016/1**

Table 5: The p-value/i results based on the Holm-Bonferroni procedure on the average influence spread results (WC model).

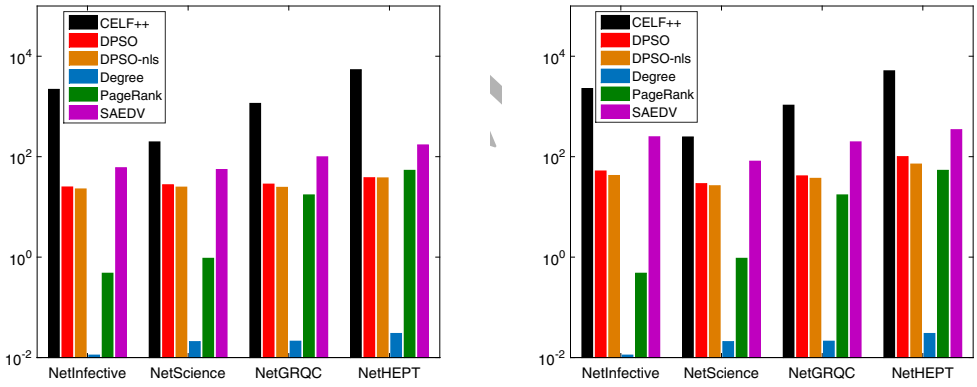
Algorithms	$k = 3$	$k = 6$	$k = 9$	$k = 12$	$k = 15$	$k = 18$	$k = 21$	$k = 24$	$k = 27$	$k = 30$
DPSO	0.4497/5	0.4497/5	0.4497/5	—	0.7055/5	0.7055/5	0.7055/5	0.4497/5	0.3717/5	—
CELF++	—	—	—	1.0/5	—	—	—	—	—	1.0/5
DPSO-nls	0.1175/4	0.0753/4	0.0752/4	0.3717/4	0.2611/4	0.1129/3	0.1763/3	0.1175/4	0.3717/4	0.6894/4
SAEDV	0.0066/1	0.0561/2	0.0701/3	0.0561/2	0.1129/3	0.1779/4	0.1763/4	0.1129/3	0.0421/3	0.3917/3
PageRank	0.0244/3	0.0701/3	0.0561/2	0.1763/3	0.0326/2	0.0561/2	0.0561/2	0.0183/2	0.0183/2	0.0934/2
Degree	0.0183/2	**0.0008/1**	**0.0008/1**	**0.0033/1**	**0.0016/1**	**0.0016/1**	**0.0016/1**	**0.0008/1**	**0.0008/1**	**0.0066/1**

The same procedure is performed to check whether there are significant differences for the WC model. The results are shown in Tables 4 and 5. Table 4 demonstrates that DPSO performs better than the remaining algorithms in term of the maximal influence spread values except that DPSO is inferior to CELF++ when $k = 9$ and $k = 24$. As for the average values of the influence spread in Table 5, DPSO can also obtain comparable values to CELF++ when $k = 12$ and $k = 30$.

The statistical tests on the IC and WC models demonstrate the proposed DPSO is comparable to the classical CELF++ algorithm, and significantly superior to the four remaining algorithms.

4.3.4. Comparisons for the running time

Apart from the comparisons of influence spread, we also compare the running time (the case of 30 seed nodes) for different algorithms in the four real-world social networks, which is shown in Fig. 7.



(a) Running time in the IC model

(b) Running time in the WC model

Fig. 7: The comparisons of running time of different algorithms in (a) the IC model and (b) the WC model.

As shown in Fig 7, CELF++ takes more than 2,000 seconds in NetInfective network. As a comparison, DPSO and DPSO-nls need almost 20 seconds. The reason why CELF++ has low efficiency in NetInfective network is that the average degree $\bar{D}$ of NetInfective social network is large and the number of edges is huge, which is as many as 17,298 accurately. The time complexity of Monte Carlo simulations is linearly related to the number of edges in the targeted network. As for the other three networks, the running times of CELF++ and PageRank increase with the size of the network. DPSO, DPSO-nls and SAEDV have stable performance on different networks in the IC model. For the WC model, the execution times of those three algorithms also tend to grow with the scale of the networks. As for the NetHEPT network in Figs. 7(a) and 7(b), CELF++ takes a few hours to select out 30 seed nodes. While, DPSO, DPSO-nls, SAEDV and PageRank can finish within 50 seconds in the IC model and 100 seconds in the WC model, respectively. Among all of the comparison algorithms, Degree has the best performance in the running time.

As summarized from the experimental comparisons, the designed local search strategy plays an important role in speeding up the convergence and finding the promising solutions in short running time. As for the influence spread,

DPSO performs better than DPSO-nls in both the IC and WC spread models. For the running time, Degree has the best performance. And PageRank also performs better than DPSO. But Degree cannot provide a seed set with good quality due to the nodes' influence overlapping problem, and Degree and PageRank show their instabilities in the influence spread. Although CELF++ greedy can provide a reliable seed set for influence maximization, it is not scalable for large-scale networks. However, the proposed DPSO algorithm can solve the problem of influence maximization effectively and efficiently in both the IC and WC models.

5. Conclusions

In this paper, we have converted the influence maximization into an optimization problem by developing a function named local influence evaluation. And then, we have desigend a novel DPSO algorithm by redefining both the representations and update rules for velocity and position to solve the influence maximization problem. A problem-specific local search strategy has been introduced to make the exploration more efficient in the decision space. The experimental results in the real social networks have shown that, compared with DPSO-nls which is a variant of DPSO, the local search can guarantee the good performance in the influence spread result and the search efficiency at the same time. Moreover, compared with the state-of-the-art CELF++ method and the other influence maximization methods, DPSO can obtain a good performance with a low time-consuming.

There are several future works that we will focus on. First, the multi-objective optimization algorithms can provide more trade-off decisions [63, 64], so we will extend our DPSO algorithm into multiple objectives to deal with the influence maximization with some constraints, e.g., budgets [43, 58] or the time of influence spread [44]. The maximization of the influence spread and the minimization of the budgets or time simultaneously can provide many choices for decision makers to promote and popularize their new products. Furthermore, we will focus on the influence maximization in multiple networks [5, 21], because the social networks are not self-governed and they are coupled and interdependent actually.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant nos. 61273317, 61422209, 61473215), the National Program for Support of Top-notch Young Professionals of China, and the Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 20130203110011).

References

[1] F. Altarelli, A. Braunstein, L. DallAsta, J. R. Wakeling, R. Zecchina, Containing epidemic outbreaks by message-passing techniques, Physical Review X 4 (2) (2014) 021024.
[2] C. Blum, X. Li, Swarm intelligence in optimization, in: Swarm Intelligence, Springer, 2008, pp. 43–85.
[3] R. M. Bond, C. J. Fariss, J. J. Jones, A. D. Kramer, C. Marlow, J. E. Settle, J. H. Fowler, A 61-million-person experiment in social influence and political mobilization, Nature 489 (7415) (2012) 295–298.
[4] S. Brin, L. Page, Reprint of: The anatomy of a large-scale hypertextual web search engine, Computer networks 56 (18) (2012) 3825–3833.
[5] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (7291) (2010) 1025–1028.
[6] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, L. Jiao, Greedy discrete particle swarm optimization for large-scale social network clustering, Information Sciences 316 (2015) 503–516.
[7] Q. Cai, M. Gong, B. Shen, L. Ma, L. Jiao, Discrete particle swarm optimization for identifying community structures in signed social networks, Neural Networks 58 (2014) 4–13.
[8] D. Chen, L. Lü, M.-S. Shang, Y.-C. Zhang, T. Zhou, Identifying influential nodes in complex networks, Physica a: Statistical mechanics and its applications 391 (4) (2012) 1777–1787.
[9] 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, ACM, 2010, pp. 1029–1038.
[10] 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, ACM, 2009, pp. 199–208.
[11] Y.-C. Chen, W.-Y. Zhu, W.-C. Peng, W.-C. Lee, S.-Y. Lee, Cim: community-based influence maximization in social networks, ACM Transactions on Intelligent Systems and Technology (TIST) 5 (2) (2014) 25.
[12] N. A. Christakis, J. H. Fowler, Connected: The surprising power of our social networks and how they shape our lives, Little, Brown, 2009.
[13] N. S. Contractor, L. A. DeChurch, Integrating social networks and human social motives to achieve social influence at scale, Proceedings of the National Academy of Sciences 111 (Supplement 4) (2014) 13650–13657.

[14] J. Derrac, S. García, 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 and Evolutionary Computation 1 (1) (2011) 3–18.
[15] 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, ACM, 2001, pp. 57–66.
[16] R. C. Eberhart, J. Kennedy, et al., A new optimizer using particle swarm theory, in: Proceedings of the sixth international symposium on micro machine and human science, vol. 1, New York, NY, 1995, pp. 39–43.
[17] R. C. Eberhart, Y. Shi, Guest editorial special issue on particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 201–203.
[18] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the american statistical association 32 (200) (1937) 675–701.
[19] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization, Journal of Heuristics 15 (6) (2009) 617–644.
[20] J. Goldenberg, B. Libai, E. Muller, Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata, Academy of Marketing Science Review 2001 (2001) 1.
[21] M. Gong, L. Ma, Q. Cai, L. Jiao, Enhancing robustness of coupled networks under targeted recoveries, Scientific reports 5 (2015) 8439.
[22] M. Gong, Y. Wu, Q. Cai, W. Ma, A. Qin, Z. Wang, L. Jiao, Discrete particle swarm optimization for high-order graph matching, Information Sciences 328 (2016) 158–171.
[23] 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, ACM, 2011, pp. 47–48.
[24] M. Granovetter, Threshold models of collective behavior, American journal of sociology 83 (6) (1978) 1420–1443.
[25] X. He, D. Kempe, Stability of influence maximization, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 1256–1265.
[26] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian journal of statistics 6 (2) (1979) 65–70.
[27] R. L. Iman, J. M. Davenport, Approximations of the critical region of the fbietkan statistic, Communications in Statistics-Theory and Methods 9 (6) (1980) 571–595.
[28] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, W. Van den Broeck, What's in a crowd? analysis of face-to-face behavioral networks, Journal of theoretical biology 271 (1) (2011) 166–180.
[29] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, K. Xie, Simulated annealing based influence maximization in social networks., in: AAAI, vol. 11, 2011, pp. 127–132.
[30] D. Kempe, J. Kleinberg, É. 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, ACM, 2003, pp. 137–146.
[31] D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, Theory of Computing 11 (4) (2015) 105–147.
[32] J. Kennedy, Particle swarm optimization, in: Encyclopedia of machine learning, Springer, 2011, pp. 760–766.
[33] J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in: Proceedings of IEEE International Conference on Computational Cybernetics and Simulation, vol. 5, IEEE, 1997, pp. 4104–4108.
[34] S. Kundu, S. K. Pal, Deprecation based greedy strategy for target set selection in large scale social networks, Information Sciences 316 (2015) 107–122.
[35] J.-R. Lee, C.-W. Chung, A query approach for influence maximization on specific users in social networks, IEEE Transactions on knowledge and data engineering 27 (2) (2015) 340–353.
[36] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Transactions on Knowledge Discovery from Data (TKDD) 1 (1) (2007) 2.
[37] 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, ACM, 2007, pp. 420–429.
[38] Y. Li, Z.-H. Zhan, S. Lin, J. Zhang, X. Luo, Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems, Information Sciences 293 (2015) 370–382.
[39] B. Liu, G. Cong, Y. Zeng, D. Xu, Y. M. Chee, Influence spreading path and its application to the time constrained social influence maximization problem and beyond, IEEE Transactions on Knowledge and Data Engineering 26 (8) (2014) 1904–1917.
[40] F. Morone, H. A. Makse, Influence maximization in complex networks through optimal percolation, Nature 524 (7563) (2015) 65–68.
[41] M. E. Newman, Finding community structure in networks using the eigenvectors of matrices, Physical review E 74 (3) (2006) 036104.
[42] M. E. Newman, Modularity and community structure in networks, Proceedings of the national academy of sciences 103 (23) (2006) 8577–8582.
[43] H. Nguyen, R. Zheng, On budgeted influence maximization in social networks, IEEE Journal on Selected Areas in Communications 31 (6) (2013) 1084–1094.
[44] Y. Ni, L. Xie, Z.-Q. Liu, Minimizing the expected complete influence time of a social network, Information Sciences 180 (13) (2010) 2514–2527.
[45] J.-P. Onnela, F. Reed-Tsochas, Spontaneous emergence of social influence in online systems, Proceedings of the National Academy of Sciences 107 (43) (2010) 18375–18380.
[46] S. Pei, L. Muchnik, J. S. Andrade Jr, Z. Zheng, H. A. Makse, Searching for superspreaders of information in real-world social media, Scientific reports 4 (2014) 5547.
[47] B.-Y. Qu, J. J. Liang, P. N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Information Sciences 197 (2012) 131–143.
[48] K. Rahimkhani, A. Aleahmad, M. Rahgozar, A. Moeini, A fast algorithm for finding most influential people based on the linear threshold model, Expert Systems with Applications 42 (3) (2015) 1353–1361.
[49] Z. Ren, H. Jiang, J. Xuan, Y. Hu, Z. Luo, New insights into diversification of hyper-heuristics, IEEE transactions on cybernetics 44 (10)

(2014) 1747–1761.
[50] K. Saito, M. Kimura, K. Ohara, H. Motoda, Super mediator–a new centrality measure of node importance for information diffusion over social network, Information Sciences 329 (2016) 985–1000.
[51] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, IEEE, 1998, pp. 69–73.
[52] G. Song, X. Zhou, Y. Wang, K. Xie, Influence maximization on large-scale mobile social network: a divide-and-conquer method, IEEE Transactions on Parallel and Distributed Systems 26 (5) (2015) 1379–1392.
[53] Y. Wang, G. Cong, G. Song, K. Xie, Community-based greedy algorithm for mining top-k influential nodes in mobile social networks, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2010, pp. 1039–1048.
[54] L. Weng, F. Menczer, Y.-Y. Ahn, Virality prediction and community structure in social networks, Scientific reports 3 (8) (2012) 618.
[55] T. Xiong, Y. Bao, Z. Hu, R. Chiong, Forecasting interval time series using a fully complex-valued rbf neural network with dpso and pso algorithms, Information Sciences 305 (2015) 77–92.
[56] W. Xu, W. Wu, L. Fan, Z. Lu, D.-Z. Du, Influence diffusion in social networks, in: Optimization in Science and Engineering, Springer, 2014, pp. 567–581.
[57] J. Xuan, H. Jiang, Z. Ren, Z. Luo, Solving the large scale next release problem with a backbone-based multilevel algorithm, IEEE Transactions on Software Engineering 38 (5) (2012) 1195–1212.
[58] Y. Yang, J. Zhang, R. Qin, J. Li, F.-Y. Wang, W. Qi, A budget optimization framework for search advertisements across markets, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 42 (5) (2012) 1141–1151.
[59] Z. Yu, C. Wang, J. Bu, X. Wang, Y. Wu, C. Chen, Friend recommendation with content spread enhancement in social networks, Information Sciences 309 (2015) 102–118.
[60] X. Zhang, J. Zhu, Q. Wang, H. Zhao, Identifying influential nodes in complex networks with community structure, Knowledge-Based Systems 42 (2013) 74–84.
[61] Y. Zhang, Q. Gu, J. Zheng, D. Chen, Estimate on expectation for influence maximization in social networks, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2010, pp. 99–106.
[62] T. Zhu, B. Wang, B. Wu, C. Zhu, Maximizing the spread of influence ranking in social networks, Information Sciences 278 (2014) 535–544.
[63] Z. Zhu, S. Jia, S. He, Y. Sun, Z. Ji, L. Shen, Three-dimensional gabor feature extraction for hyperspectral imagery classification using a memetic framework, Information Sciences 298 (2015) 274–287.
[64] Z. Zhu, J. Xiao, S. He, Z. Ji, Y. Sun, A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one dynamic pickup-and-delivery problem, Information Sciences 329 (2016) 73–89.