Journal of Network and Computer Applications 103 (2018) 119–130

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

DDSE: A novel evolutionary algorithm based on degree-descending search strategy for influence maximization in social networks

Laizhong Cui[a], Huaixiong Hu[a], Shui Yu[b,*], Qiao Yan[a], Zhong Ming[a], Zhenkun Wen[a], Nan Lu[a]

[a] *College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, PR China*
[b] *School of Information Technology, Deakin University, Melbourne, Australia*

ARTICLE INFO

Keywords:
Influence maximization
Evolutionary algorithm
Heuristic method
Social network

ABSTRACT

Influence maximization (IM) is the problem of finding a small subset of nodes in a social network so that the number of nodes influenced by this subset can be maximized. Influence maximization problem plays an important role in viral marketing and information diffusions. The existing solutions to influence maximization perform badly in either efficiency or accuracy. In this study, we analyze the causes for the low efficiency of the greedy approaches and propose a more efficient algorithm called degree-descending search evolution (DDSE). Firstly, we propose a degree-descending search strategy (DDS). DDS is capable of generating a node set whose influence spread is comparable to the degree centrality. Based on DDS, we develop an evolutionary algorithm that is capable of improving the efficiency significantly by eliminating the time-consuming simulations of the greedy algorithms. Experimental results on real-world social networks demonstrate that DDSE is about five orders of magnitude faster than the state-of-art greedy method while keeping competitive accuracy, which can verify the high effectiveness and efficiency of our proposed algorithm for influence maximization.

1. Introduction

With the popularity of booming internet technology, billions of people are involved in and connected through social networks (Zhou et al., 2017; Han et al., 2016), such as Facebook, Twitter and LinkedIn. Generally speaking, a social network is a structure connecting people or organizations for communications. There are many kinds of social networks such as email networks, phone contact networks, online social networks (Haralabopoulos et al., 2015), mobile social networks (Lu et al., 2014) and collaboration networks of scientists (Kimura et al., 2006) etc. Connecting billions of people and generating tons of data every day (Cui et al., 2016), social networks are not just communication channels but also great platforms for news propagation, public services and especially commercial advertising (Bond et al., 2012, Contractor et al., 2014, Onnela et al., 2010),even used as forensics tools recently (Quick and Choo, 2017). Social networks can generate massive data so that we need the big data technologies (Lv et al., 2017) to handle the issues including data types, storage models, data privacy and security. More and more researchers are focusing on using machine learning techniques (Meng et al., 2016) to analyze social network data and discover models and patterns (Zhang et al., 2017) of the people's behaviors in social networks, such as crowd-sensing(Sun et al., 2014) and friends or purchase recommender systems (Wu et al., 2013; Wang et al., 2015). Previous researches show that the “word-of-mouth” effect (Bass, 1976; Brown et al., 1987; Domingos et al., 2001; Goldenberg and Libai, 2001; Goldenberg et al., 2001; Mahajan et al., 1990; Richardson et al., 2002) plays a very important role in spreading innovations and ideas in social networks. People tend to trust and accept new products or ideas that come from their families, friends (Bryant et al., 2004). When social networks are used for commercial promotion, with a limited budget, the marketing staffs of a start-up company can only select a small number of initial users and let them try their products. Targeting a small group of the most influential people in available social networks and propagating their influence to as many potential customers as possible is critical for viral marketing. This problem is formally called as influence maximization.

Influence maximization is defined as finding a K-sized subset of users (usually called seeds, where the number K can be specified in advance) that can lead to the maximum influence spread in social networks.

Kempe et al. (2003) formulated the influence maximization as a discrete optimization problem and proved that the influence maximization is NP-hard. Moreover, they proposed a natural hill-climbing greedy algorithm with a provable approximation ratio of (1-1/e). Their naive greedy approach is significantly better in accuracy compared with the random method and degree based heuristics. However, their greedy

[*] Corresponding author.
E-mail address: syu@deakin.edu.au (S. Yu).

https://doi.org/10.1016/j.jnca.2017.12.003
Received 16 July 2017; Received in revised form 27 September 2017; Accepted 2 December 2017
Available online 14 December 2017
1084-8045/ © 2017 Elsevier Ltd. All rights reserved.

algorithm performs very poorly in efficiency, especially for the large social networks including tons of nodes. In recent years, many greedy approaches that involve efficiency enhancement techniques showed up. Leskovec et al. (2007) proposed an algorithm called CELF that can speed up the node-selecting process by exploiting the sub-modularity property. The CELF algorithm is 700 times faster than the naive greedy. Chen et al. (2009) proposed a novel greedy algorithm called NewGreedy, which improved the efficiency by simplifying the network structures. Wang et al. (2010) utilized the community property of social networks and proposed a community-based algorithm named CGA. Goyal et al. (2011) proposed CELF++ and demonstrated it is 35–55% faster than CELF.

The simple greedy algorithm has the following limitations: (1) it requires repeated computation of the marginal influence spread of each candidate node in the node selecting process. According to previous researches, the computation of spread under both IC and LT model is #P hard (Chen et al., 2010, 2011); (2) the computation of the marginal influence spread of each node requires tens of thousands of Monte Carlo simulations to obtain an accurate estimate, which is very time consuming. Recent efficiency-enhanced greedy algorithms have made notable improvements to overcome the first or second limitation, but they cannot do both. Low efficiency is still the most significant defect of greedy approaches.

In this paper, we propose a novel evolutionary algorithm called degree-descending search evolution (DDSE) for the influence maximization in social networks. The basic idea is to eliminate repeated simulations and replace it with rough estimation. Our evolutionary algorithm includes an objective function named expected diffusion value (EDV) (Jiang et al., 2011) for the estimation. The DDSE algorithm tackles the efficiency issue from the following aspects: (1) EDV estimates the influence spread of the seed set in a local area of each node with an equation that can be computed instantly. Therefore it avoids the repeated simulations, which are widely used in greedy approaches; (2) DDSE calculates EDV, i.e. the overall local influence spread of the candidate seed set simultaneously, instead of computing the marginal influence of the candidate node one by one during the selecting process like greedy algorithms do; (3) Efficient evolutionary operators based our degree-descending search strategy make the DDSE algorithm only requires a very little population which means very few computations are involved in the evolution process.

The proposed algorithm has two notable features: Firstly, it is very efficient and it significantly outperforms the state-of -art greedy algorithm in efficiency. Second, it has a very good accuracy and performs much better than the existing heuristics. In summary, our contributions could be summarized as follows:

1. We propose a new degree-based heuristic method called degree descending search (DDS), which is able to generate a solution with the competitive quality compared with the traditional degree centrality heuristic method.
2. Based on the strategy of DDS and EDV, we develop a novel evolutionary algorithm called degree-descending search evolution (DDSE), which overcomes the efficiency issue of greedy approaches by avoiding repeated simulations in the node selecting process.
3. We conducted extensive experiments on three real-world social networks. Experimental results demonstrate that our algorithms is five orders of magnitude faster than the state-of-art greedy method CELF, while keeping competitive diffusion spread and beating other representative approaches in terms of both accuracy and efficiency.

The rest of this paper is organized as follows. In Section 2, the related works for influence maximization is introduced. In Section 3, we state the preliminaries including diffusion models, problem definition and a brief introduction to evolutionary algorithms. Our proposed algorithm is presented in Section 4. Experiments results and analysis are described in Section 5 and we conclude this paper in Section 6.

2. Related works

The influence maximization problem is first proposed by Domingos and Richardson (2001). They regarded the influence maximization as an algorithmic problem and proposed a probabilistic solution. Kempe et al. studied the influence maximization in a dynamic manner. They investigated two most widely studied models for the influence diffusion process, i.e. independent cascade (IC) model (Goldenberg et al., 2001) and linear threshold (LT) model (Granovetter et al., 1978; Rahimkhani et al., 2015), that describe how users of social networks propagate their influences to their friends. Based on these two models, they formulate the influence maximization as a combination optimization problem and proved that influence maximization is a NP-hard optimization problem. They also proposed a natural hill-climbing greedy algorithm and they proved that their simple greedy method has an approximation ratio of (1-1/e). Experimental results show that their approach is much better in accuracy compared to traditional random method and degree based heuristics.

Ever since that, greedy approaches become the mainstream of researches on the influence maximization. However, the great limitation of naive greedy method proposed by Kempe et al. (1) is its inefficiency, especially for the large social networks including tons of nodes. With their greedy algorithm, finding a small seed set in a moderately large network (e.g. 15,000 vertices) need take a few days to complete on a modern server machine, which is unbearable for people. The essential problem of their method lies on the fact that under IC diffusion model the exact influence spread of a set of nodes cannot be easily computed, and the value can only be approximated by running repeated Monte Carlo simulations (Kempe et al., 2003). To achieve a reliable value of influence spread of a node set, Monte Carlo simulation needs to be run tens of thousands of times, which is very time-consuming. The analysis results can tell us that the simple greedy algorithm has to do that simulations for O (N*K) nodes to find out the top-K influential nodes for a social network containing N nodes. That's why the greedy algorithm runs so slowly.

In recent years, to enhance the efficiency of the greedy algorithm, some considerable works have been done. Some researchers try to solve the problem by cutting down unnecessary computations in the node selecting process. Leskovec et al. (2007) improved the naive greedy by exploiting the sub-modularity that shows up in the node-selecting process, i.e. the marginal influence spread of a target node is less when it is added into a larger rather than smaller seed set $\boldsymbol{S}$. By utilizing the sub-modularity property, they developed an algorithm called CELF, whose main idea is an optimization called "lazy-forward" while selecting the most influential node. Their optimization efficiently reduces the number of evaluations on marginal influence spread of each node by removing unnecessary expensive Monte Carlo simulations. Experimental results show that their method is 700 times faster than a simple greedy algorithm while achieving near optimal solution. Chen et al. (2009) proposed a faster greedy algorithm called NewGreedy. The main idea of this algorithm is to remove the edges that do not contribute to the propagation of influence from the original large graph to get a smaller graph and do the diffusion on the smaller graph. Experimental results show that NewGreedy significantly outperforms CELF. Wang et al. (2010) improved the efficiency by exploiting the community property (Arab et al., 2014) of social networks and proposed a community-based algorithm named CGA, which is applicable to both IC model and LT model. Moreover, Goyal et al. (2011) proposed CELF++, and demonstrated it is 35–55% faster than CELF. Heidari et al. (2015) proposed a scalable algorithm called State Machine Greedy Algorithm (SMG), which reduces the calculations of counting the traversing nodes in estimate propagation procedure and simplifies the Monte Carlo graph construction in simulation of diffusion. Experiments demonstrate that their approach is faster than CELF.

On the other hand, some researchers studied the influence maximization by composing new efficient heuristics. Degree heuristic (Kempe et al., 2003) is one of the most commonly used heuristics for selecting seeds in influence maximization. The basic idea of degree heuristics is that the nodes with larger degree have larger influence spread. This simple idea conforms to our intuitions. Experimental results show that the Degree heuristic outperforms the Random method. Distance based methods (Kempe et al., 2003; Chen et al., 2009) are also applicable to the influence maximization. The distance based methods calculate the average shortest path length of each node to all other nodes, and the top K nodes with the minimum average shortest path are firstly selected into the seed set. PageRank algorithm (Brin et al., 2012) is created to evaluate the importance of web pages. But now it is also used influence maximization. Similar to web pages, PageRank evaluates the importance of a node by computing its PageRank score. Recently, Chen et al. (2009) proposed a new heuristic named Degree Discount, which is based on an interesting fact that most central nodes may be clustered. Therefore, selecting all of them as seeds is actually not necessary. After selecting a node with a large degree, the degree of its neighbors should be discounted. Then the node with the largest degree after discounting is selected as a seed. Experimental results show that Degree Discount outperforms other heuristics including degree centrality, distance-based method and random approach. In addition, Chen et al. (2010) proposed a new greedy heuristic called maximum influence arborescence (MIA) for influence maximization. They build up a local arborescence structures for each node based on its maximum influence paths (MIP) to other nodes. After that, they evaluate the local influence of each node in that arborescence structures and use the local influence to estimate the global influence spread. Greedy strategy is applied subsequently. Experiments show that their method is fast and scalable for large social networks. In summary, heuristics are very efficient and scalable for large social networks, but they are not preferred because they are not guaranteed in accuracy and they are not robust with respect to network structures.

Moreover, many researchers devoted to solve the influence maximization problem by taking advantage of the inherent properties of the social network, e.g. community structures. Wang et al. (2010) utilized the community property of social networks and proposed a community-based algorithm named CGA. Their algorithm splits the network into several communities adaptively and then employs dynamic programming to select communities for finding the most influential nodes.

It is very difficult to solve NP-hard problems and achieve the satisfying results with the common methods. While the computation intelligence methods such as Genetic Algorithms, Particle Swarm Optimization, sometimes can perform surprisingly well in handling NP-hard optimizations. Inspired by this observation and the work of Jiang et al. (2011), we design our approach as a swarm intelligent algorithm, in which we avoid the expensive simulation by integrating estimation function and thus tacking the efficiency problem of existing greedy algorithms successfully.

3. Preliminaries and problem statement

In this section, we first list the notations that will be used in the subsequent sections. And we introduce two widely used diffusion models and formulate the definition of influence maximization. After that, we give a brief introduction to the evolutionary algorithms since our algorithm is designed as an evolutionary algorithm.

A social network can be denoted as a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the set of nodes in this network and $\boldsymbol{E}$ is the set of edges between the nodes. In a graph $\boldsymbol{G}$, a node represents an individual and an edge represents the social relationship (collaboration, trust, etc.) between two individuals. In social networks, the individuals could influence each other through the interactions between them. The nodes (individuals) in social networks are either active or inactive. If an active node $\boldsymbol{u}$ spreads an idea or innovation to node $\boldsymbol{v}$ and node $\boldsymbol{v}$ accepts it, we can say node $\boldsymbol{v}$ is activated and become active. Otherwise, node $\boldsymbol{v}$ is inactive. The dynamic process of interactions between nodes represents the diffusion of influence. To describe the interaction process and behavior paradigm of nodes, we need a diffusion model.

3.1. *Diffusion model*

There are many diffusion models that describe the way of interactions and information propagation between users in social network, such as epidemic model (Peng et al., 2014; Wen et al., 2017) and cascade model. Our algorithm focuses on the influence maximization under the independent cascade (IC) model. In the **independent cascade model** (Goldenberg et al., 2001; Kempe et al., 2003), the state of a node in a social network is either active or inactive. A random active node $\boldsymbol{u}$ has only one chance to activate its inactive neighbor node $\boldsymbol{v}$. When node $\boldsymbol{u}$ tries to activate its neighbor $\boldsymbol{v}$, node $\boldsymbol{v}$ may be activated and become active with a pre-specified probability (called active probability) $\boldsymbol{p}$ or stay inactive with probability (1-$\boldsymbol{p}$).

If the activation succeeds, node $\boldsymbol{v}$ will stay active and it is capable of activating its own neighbors as well. Otherwise, node $\boldsymbol{v}$ can only be activated by its another neighbor and node $\boldsymbol{u}$ can never activate $\boldsymbol{v}$ again. The diffusion process will stop when no more new nodes are activated.

3.2. *Problem statement*

Influence spread is defined as the expected number of active nodes when the influence diffusion process in a specific diffusion model initiated by a specified node set S (called seeds) stops. The influence spread of a given node set S can be denoted as $\sigma(S)$. The influence maximization is a problem about how to find out a node set S so that its influence spread $\sigma(S)$ is maximum.

Based on the above descriptions, the influence maximization is a constrained optimization problem and it can be defined as follows.

Given a graph $G = (V, E)$ and a number K ($0 < K < |V|$), select K nodes as the initial node set $S = \{s_1, s_2, ... s_K\}$ and let them propagate their influence in a specific diffusion model, so that the final influence spread $\sigma(S)$ is maximum. This optimization problem can be formulated as Eq. (1).

$$\begin{cases} S = \arg\max \sigma(S) \\ subject \quad to |S| = K \end{cases} \tag{1}$$

The computation of the influence spread of a given node set is #P-hard (Zhang et al., 2010; Lee et al., 2015) and Kempe et al. (2003) proved that the influence maximization is a NP-hard optimization problem. Therefore, it's very challenging to solve the influence maximization with the strict time constraint.

3.3. *Evolutionary algorithm*

Evolutionary algorithms (EAs) (Back, 1996) are a class of stochastic search algorithms that can be applied to both combinational and numerical optimization problems. The major representatives of evolutionary algorithms are evolution strategies (ESes), evolutionary programming (EP) and genetic algorithms (GAs).

Inspired by natural biological evolution processes, evolutionary algorithms adopt many biological and genetic concepts such as populations of individuals, and operations such as selection, recombi-

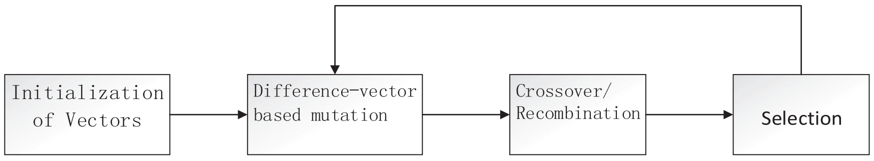


Fig. 1. Framework of DE.

nation and mutation in their paradigms for optimization problem solving.

In the paradigm of solving optimization problems with evolutionary algorithms, we need to determine the solution space and establish an objective function for the problem. After that evolutionary operations (i.e. selection, recombination and mutation) are utilized iteratively to find a Pareto optimal solution. For example, differential evolution (DE) (Das et al., 2011) algorithm, which is a swarm intelligent algorithm, works in a paradigm like the following:

As shown in Fig. 1, DE is constructed with four components (also can be called operators): initialization, mutation, crossover or recombination and selection. What's notable about DE is that there are parameters mutation operator and crossover operator to control the quality of final results: mutation probability f for mutation operator and crossover probability cr for crossover operator.

4. Proposed algorithm

In this section, we give the overall framework and details of the proposed Degree Descending Search Evolution (DDSE) algorithm for influence maximization in the social network. But firstly, let us fix the representation issue. As we know, the influence maximization aims to find a node set that generate the maximum influence spread. And our proposed method is designed as an evolutionary algorithm (a swarm intelligence algorithm, to be more specific). Therefore we have to decide how to represent the solutions of the IM problem in our algorithm.

4.1. Representation

In our proposed algorithm, each individual of the population is a K-sized seed set and it is represented by a K-dimensional integer vector $\boldsymbol{X_i} = (x_{i1}, x_{i2}, \ldots, x_{ik})$, where each element $x_{ij}(j = 1,2,\ldots,K)$ of $\boldsymbol{X_i}$ is a node in the graph G.

For example, given $K = 5$ and vector $\boldsymbol{X_i} = (1,5,12,34,56)$, it means we have chosen the node 1,5,12,34 and 56 from the network G as the seed set for the influence maximization problem.

4.2. Framework of Degree Descending Search Evolution

The framework of DDSE comprises four primary components, i.e. initialization, mutation, crossover and selection. Fig. 2 gives an overview of the proposed evolution algorithm for influence maximization.

As shown in Fig. 2, our proposed algorithm is composed just like the framework of the Differential Evolutionary (DE) algorithm—both DE and our DDSE have four components, i.e. initialization, mutation, crossover and selection. But DDSE adds the degree descending search strategy (DDS) in the framework. DDS is integrated with all four primary components and that's why our algorithm can work efficiently. Let's explain our proposed algorithm briefly as follows.

In the initialization, the initial population $\boldsymbol{X}$ **is generated**, where each individual is a node set representing a feasible solution for the influence maximization problem;

1) Mutation procedures are used to make disturbance to the original population and create new solutions. More specifically, the mutation procedures make changes to each individual of the original population $\boldsymbol{X}$. After the mutation procedures, the population is mixed up with a small group mutated individuals and becomes a new population $\boldsymbol{X}$ *(mutation)*;
2) Crossover operations generate a mixed-up population filled with both original individuals and mutated individuals.
3) Selection procedures incorporate the greedy strategy and make the comparison between the fitness value of each individual of $\boldsymbol{X}$ *(mutation)* and that of individual of $\boldsymbol{X}$. The winner (the one with better fitness) is selected into the population of the next generation.
4) Then the population of the new generation is mutated again in the next round and the process goes on. The whole process is stopped if the stop criteria are met. The individual with the best fitness is the desirable solution for the influence maximization.

Note that our degree descending search strategy is employed in initialization, mutation and crossover operators. To represent our algorithms more clearly, we describe the DDSE algorithm with the pseudo codes in Algorithm 1.

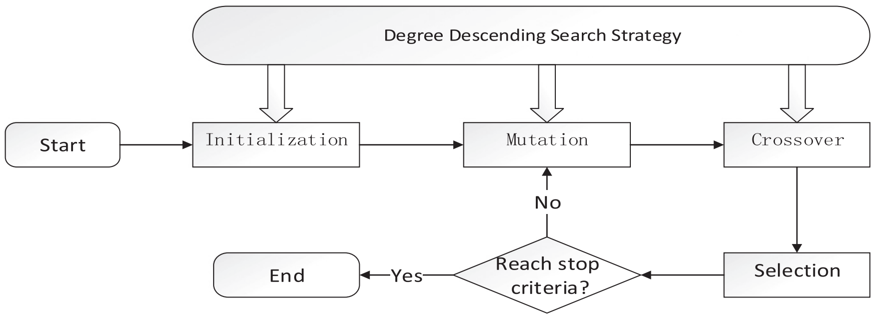


Fig. 2. Overall framework of DDSE algorithm.

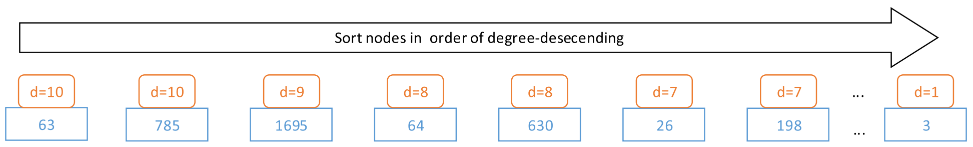


Fig. 3. Degree Descending Search.

Algorithm 1. Framework of Degree Descending Search Evolution algorithm for the influence maximization

```
Input: Graph $G = (V,E)$, the size of population $n$, the number of
    iterations $g_{max}$, mutation probability $f$, crossover probability $cr$
    and the size of seed set $K$.
Step 1 Initialization:
  Step 1.1 Initialize the population: $X$ =
  Degree_Descending_Initialization ($n$, $K$)
  Step 1.2 Set iteration counter to zero: $g = 0$
Step 2 Mutation:
1: Initilize $X$ (mutation) with empty vectors;
2: Sort all nodes of $V$ in degree-descending order;
3: for each $i < = n$ do
4:    $Xi$ (mutation) = $Xi$;
5:    up-bound = $K*(i+5)$;
6:    if up-bound > $|V|$ then
7:    up-bound = $|V|$;
8:    end if
9:    for each element $Xij$ (mutation) $\in Xi$ (mutation) do
10:     if random < $f$ then
11:     $Xij$ (mutation) = $V$. get (random (up-bound));
12:     end if
13:     end for
14: end for
Step 3 Crossover:
1: Initialize $X$ (crossover) with empty vectors;
2: Sort all nodes of $V$ in degree-descending order;
3: for each $i < = n$ do
4:    $Xi$ (crossover) = empty;
5:    up-bound = $k*(i+5)$;
6:    if up-bound > $|V|$ then
7:    up-bound = $|V|$;
8:    end if
9:    for each $Xij$ (crossover) $\in Xi$ (crossover) do
10:     if random < $cr$ then
11:     node = $Xij$ (mutation);
12:     while node is in $Xi$ (crossover)
13:     node = $V$. get (random (up-bound));
14:     end while
15:     else
16:     node = $Xij$;
17:     while node is in $Xi$ (crossover)
18:     node = $V$. get (random (up-bound));
19:     end while
20:     end if
21:     $Xij$ (crossover) = node;
22:     end for
23: end for
Step 4 Selection:
  Initialize $X$(selection) with empty vectors;
  for each $i < = n$: $Xi$(selection) = arg max{ $EDV(Xi)$, $EDV(Xi$(crossover)) };
Step 5 Update $X$ and continue
  Step 5.1 $X = X$ (selection);
  Step 5.2 Determine to stop or continue:
  if $g < g_{max}$, go to step 2 and set $g = g+1$;otherwise, go to Step 6;
Step 6 Local search:
  Step 6.1 $Gbest = arg\ max$ {EDV ($Xi$)};
  Step 6.2 $Gbest = Local\_Search(Gbest)$;
Output: $Gbest$, which represents the solution with maximum
  influence spread.
```

In our algorithm, we employ two techniques that are critical for improving the efficiency: the objective function EDV and our proposed Degree-Descending Search strategy. The EDV function helps to remove the traditional repeated simulations required to estimate the influence spread of target node set. And our proposed Degree-Descending Search strategy are utilized create effective and efficient evolution operators, i.e. the mutation operator and the crossover operator. Let introduce them in details in the following sections.

4.3. Objective function

Jiang et al. (2011) proposed a metric called expected diffusion value (EDV) to estimate the actual influence spread under IC model and employed it in their algorithm to replace expensive simulations. The EDV approximates the spread by computing how many direct neighbors of the nodes in the candidate node set **S** are expected to be activated, which can be described as follows,

$$\mathrm{EDV}(S) = k + \sum_{i \in N_S^{(1)}/S} (1 - (1 - p)^{\tau(i)}) \tag{2}$$

where $N_S^{(1)}$ represents the direct neighbors of the node set S, and $i \in N_S^{(1)}/S$ represents that node i belongs to $N_S^{(1)}$ but it is not in the set S. p is the active probability in IC model and $\tau(i)$ is the number of the links between node i and the set S.

4.4. Degree-descending search (DDS)

We propose a degree based heuristic called degree-descending search, which is utilized in the evolutionary operators including mutation and crossover to speed up the convergence of our evolutionary algorithm. Like the degree centrality method, the main idea of degree-descending search is that a node with the larger degree tends to be more influential. However, the results of previous works (Kempe et al., 2003; Chen et al., 2009) demonstrate that the node set consists of top-K nodes with the largest degree cannot reach the widest influence spread. Therefore, we seek the nodes with the larger degree, but we do not require the nodes have to be with the largest degree. That is the main difference between the degree centrality method and our degree descending search – the latter has a looser requirement for the nodes' degree.

Assume we are given a graph G which has the following nodes: 785, 63, 1695, 630, 64, 198, 26, 3 and etc. As shown in Fig. 3, the traditional degree centrality method for top-K influence maximization will generate the solution S = {63, 785, 1695, 64, 630} when K = 5. In contrast, our degree descending search heuristic may generate many solutions, such as S1 = {63, 785, 1695, 630, 26}, S2 = {63, 1695, 64, 630}, S3 = {785, 1695, 64, 630, 26}, etc. since the degree descending search strategy seeks nodes with large but not always the largest degrees.

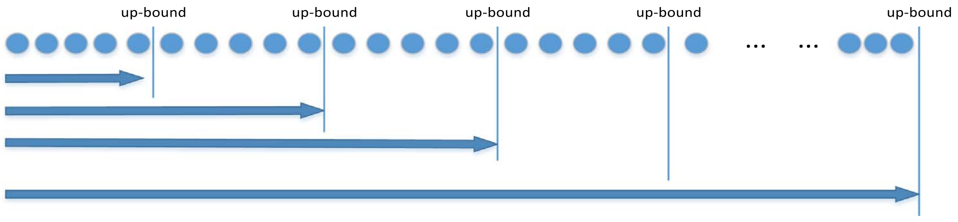


Fig. 4. Expanding search range of DDS strategy.

Algorithm 2 gives the pseudo code of our degree descending search.

Algorithm 2. Degree_Descending_Search(G,K):

```
1: Input: Graph $G = (V, E)$, the size of seed set $K$
2: Set $S$ = empty
3: Sort all nodes of $V$ in degree-descending order
4: for each $i <= K$ do
5:     up-bound = $i+5$
6:     node = $V$.get (random (up-bound))
7:     $S$.add (node)
8: end for
9: Output: node set $S$
```

As shown in Algorithm 2, our degree-descending search sorts all nodes in degree-descending order and the nodes with the larger degree are first selected. But we do not simply choose the node with the largest degree but randomly choose a node whose degree is within a specific range (line 5) and the range is adjustable and controlled by a flexible up-bound. With the increasing up-bound, the range for node-selection expands gradually and all nodes in the social network G are covered eventually. Meanwhile, the nodes with the larger degrees are selected with priority. Fig. 4 illustrates the expanding search range of DDS where nodes with large degree have priorities. With this mechanism, we can obtain a node set that is comparably influential as what the degree centrality method gets while reserving the flexibility of the solution. This flexibility is critical as it makes it possible for our algorithm to generate new feasible solutions in the mutation and crossover phase.

4.5. Initialization based on DDS

To make our evolutionary algorithm converge fast and achieve a better solution, we initialize a diversified population with our degree descending strategy rather than randomly pick some nodes. The main procedures are described in Algorithm 3.

Algorithm 3. Degree_Descending_Initialization (G, n, K, div)

```
1: Input: Graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, the size of population $n$, the size of
   seed set $\boldsymbol{K}$, diversity degree $\boldsymbol{div}$
2: Sort all nodes of $\boldsymbol{V}$ in degree-descending order
3: for each $i <= n$ do
4:     $\boldsymbol{Xi}$ = Degree ($G$, $K$);
5:     up-bound = $K*(i+5)$;
6:     if up-bound > $|V|$ then
7:     up-bound = $|V|$;
8:     end if
9:     for each element $\boldsymbol{Xij} \in \boldsymbol{Xi}$ do
10:      if random < $div$ then
11:      $\boldsymbol{Xij}$ = $V$.get(random(up-bound));
12:      end if
13:    end for
14: end for
15: Output: the initial population $\boldsymbol{X}$.
```

As shown in Algorithm 3, all vectors are firstly initialized with the degree based method, which selects K influential nodes with the largest degree in graph G. This procedure is denoted in line 2 and Degree (G, K) in line 4.

However, this simple degree heuristic method will lead to a population with poor diversity. To overcome this problem, we replace each element of vectors with a random node that is different from other elements of the vector. Note that we have to make sure that the random node generated in line 11 is unique in $\boldsymbol{Xi}$.

We can control the diversity of the population with an adjustable diversity parameter value, e.g. 0.3. We use a random number between interval 0 and 1 for the determination. If the generated random number is less than the diversity parameter value, that element of the vector will be replaced by a random node whose degree is within a controllable up-bound. The flexible range of degree of the nodes reflects the essential idea of our degree descending search strategy.

For example, 10 nodes with the largest degree in graph G are: 63, 785, 64, 1695, 26, 1834, 1841, 1696, 630, and 1032, respectively. With Degree (G, 5), we have the vector $\times_1$ = (63, 785, 64, 1695, 26) and we generate a random value $\boldsymbol{r}$ for each element $\times_{1j}$.

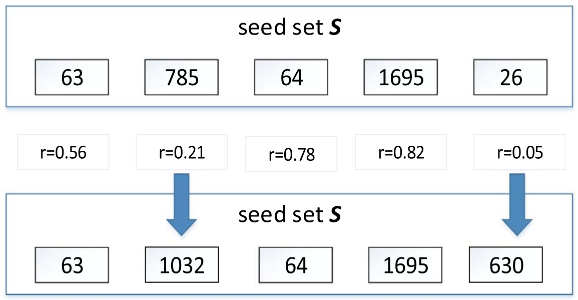


Fig. 5. Degree-Descending initialization.

Fig. 5 illustrates how we generate a new vector, where the diversity degree value is 0.3 and $K = 5$.

As shown in Fig. 5, during the initialization process, the element $\boldsymbol{X}_{\text{ij}}$ is replaced with another unique node with a large degree if the generated random value is less than the value of the diversity parameter.

The degree based initialization in Algorithm 2 can be completed quickly. After the degree based initialization, a population with good diversity can be obtained. The good diversity can help the algorithm to achieve a solution with better influence spread.

4.6. Mutation operator

Mutation operator is introduced to generate new individuals so that the population can cover a wide range of feasible solutions during the evolution process, making it possible to find the global optimal or near-optimal solution to the influence maximization problem. The mutation operator utilizes a random number and a pre-specified probability f to determine each element of the original individual mutate or not: if the random number generated is less than f, the element is replaced with a unique node generated by the DDS; otherwise it stays the same. Our mutation rules can be described with Eq. (3).

$$Xij(Mutation) = \begin{cases} Xij & if \quad rand >= f \\ uniq(DDS(i, K, G)) & if \quad rand < f \end{cases} \tag{3}$$

The detailed pseudo codes for the implementation of mutation rules are given in the Mutation part of Algorithm 1. Note that we have to make sure that the random node generated by DDS strategy in line 11 is unique in ***Xi** (mutation)*.

4.7. Crossover operator

After the vectors are mutated, we make cross-over operations between original population ***X*** and mutated population ***X** (mutation)* to generate a new mixed population for subsequent selections. Algorithm 5 gives the details of the crossover operation. The crossover operator utilizes a random number and a pre-specified probability *cr* to determine the value of the *X(crossover)*:if the random number generated is less than *cr*, the i-th individual *Xi(crossover)* of *X(crossover)* will be filled with *Xij(mutation)* of *Xi(mutation)* (or an unique node generated by the DDS if *Xij(mutation)* is already in the *Xi(crossover)*);otherwise it will be filled with *Xij* of the original individual *Xi* (or an unique node generated by the DDS if *Xij* is already in the *Xi(crossover)*). Our crossover rules can be described with Eq. (4).

$$Xij(crossover) = \begin{cases} Xij \;\; or \;\; uniq(DDS(i, K, G)) & if \;\; rand >= cr \\ Xij(mutation) \;\; or \;\; uniq(DDS(i, K, G)) & if \;\; rand < cr \end{cases} \tag{4}$$

As shown in crossover part of Algorithm 1, the codes on line 2-line 8, line 13 and line 18 suggest that our degree descending strategy is utilized to generate new nodes for the update of ***Xij** (crossover)*. Line 11 and line 16 suggest that the population after the crossover operation is a combination of the original population ***X*** and the mutated population ***X** (mutation)*. Line 12- 14 and line 17–19 make sure that the elements are unique in each crossover individual.

4.8. Selection

We employ the greedy strategy in selection procedures. As described in Eq. (5), each individual of original population ***X*** is compared with the individual of ***X** (crossover)*. The one with the larger EDV wins and it is selected into seeds. The details are described with the pseudo codes in Algorithm 1.

$$Xi(selection) = \max\{EDV(Xi), EDV(Xi(crossover))\} \tag{5}$$

4.9. Local search

After generations of evolution, we can find a solution with the good quality, but we further utilize a neighborhood-based optimization–local search (proposed in the previous work by Gong et al., 2016), to improve it for better influence spread before updating the population ***X***. The main idea of the local search optimization is to replace each element of the node set *S* with one of its neighbors. If the new seed has a larger EDV, we accept the replacement. Otherwise, we keep the original element. Algorithm 4 gives the pseudo codes of the local search.

Algorithm 4:. Local search

```
1: Input: node set S.
2: S' = S;
3: for each node u in S do
4:     for each node v in Nb (u) do
5:     St = Replace(S', u, v);
6:     if EDV (St) > EDV(S') then
7:     S' = St;
8:     end if
9:     end for
10: end for
11: Output: node set S'
```

In the above pseudo codes, *Nb (u)* in line 4 indicates the neighbors of node ***u***. Replace(***S', u,v***) means replacing the node *u* of *S'* with its neighbor *v*.

4.10. Computation complexity analysis

We analyze the computational complexity of the proposed DDSE algorithm according to its process described by all the aforementioned pseudo codes of Algorithm 2 to Algorithm 7. Here, we use n, k, g_{max} to denote the population size, node set size and the number of iterations, respectively. For **Step1.1** of the initialization process in Algorithm 2, it needs $O\ (n\cdot k)$ basic operations. After the iterations begin, both the **Step2 Mutation** and **Step3 Crossover** operator have a complexity of $O\ (n\cdot k)$. **Step4 Selection** operator has a complexity of $O\ (n)$. Updating of the population requires $O\ (1)$ operations. Local search of **Step 6.2** requires $O\ (k\cdot D(\text{—}))$ basic operations. Moreover, **Step4** and **Step 6.2** include the calculation of EDV value, which has the time complexity of $O\ (k\cdot D(\text{—}))$. To sum up, the overall time cost of DDSE algorithm is $O\ (n\cdot k)$ + $[2\ O(n\cdot k)+O(n\cdot k\cdot\ D(\text{—}))+O(1)]^*\ g_{max}+\ O(k^2\cdot D(\text{—})^2)$. Therefore, the time complexity of DDSE is $O\ (n\cdot k\cdot D(\text{—})\cdot g_{max})$ according to the rules of symbol O.

5. Experiments

We conduct experiments on real-world social networks to evaluate the effectiveness and efficiency of the proposed DDSE algorithm. The data sets, experimental setup and comparison algorithm are first given. Then, we perform experiments to determine the best value for parameters involved in DDSE algorithm. Finally, the experimental results of all comparison algorithms are compared and discussed in terms of influence spread and running time.

5.1. Experiment setup

5.1.1. Algorithms & data sets

Four well known algorithms for influence maximization are taken into the comparison with our algorithm DDSE, i.e. CELF greedy algorithm (Leskovec et al., 2007), PageRank (Brin et al., 2012), Degree (short for degree centrality, Kempe et al., 2003) and SAEDV (Jiang et al., 2011). We give the main features and descriptions of all involved algorithms in Table 1.

In our experiments, to make it more convincing, we make use of three real-life scientific co-author networks representing the cooperative relationships between scientists. All of the data sets are cited from previous studies, where the authors extract them from paper-lists in sections of the e-print arXiv. Among all these data sets, NetScience is constructed with 1461 nodes (Newman et al., 2006) by taking names of scientists who co-authored papers in condensed-matter section of physics on arXiv. NetGRQC is extracted from the "General Relativity and Quantum Cosmology" (Gr-Qc) section with 5242 nodes and 28,980 edges (In: http://snap.stanford.edu/data/ca-GrQc.html). The network NetHEP is from the "High Energy Physics - Theory" (Hep) section with 15,233 nodes and 58,891 edges (Chen et al., 2009). In these social

Table 1
The features and descriptions of all comparison algorithms.

Algorithms	Features	Descriptions
CELF	Lazy forward	Greedy algorithm
DDSE	Degree descending search	Evolutionary approach
PageRank	Measure importance by votes	Simple heuristic
Degree	The node with more neighbors are influential	Simple heuristic
SAEDV	Simulated annulling, estimation with EDV	Evolutionary approach

Table 2
Statistic characteristics of three real-world social networks.

Data set	Nodes	Edges	Average Degree
NetScience	1461	2742	1.88
NetGRQC	5242	28,980	5.53
NetHEP	15,233	58,891	3.87

Table 3
Parameters of DDSE algorithm.

Parameters	**Descriptions**
f	Mutation probability
cr	Crossover probability
g_{max}	Number of generation the evolution lasts
n	The size of the population
$diversity$	The diversity of the population

networks, each node represents an author of co-written paper and the number of edges between a pair of nodes is equal to the number of papers the two authors collaborated. All these data sets are used in previous researches. Table 2 is the characteristics overview of the three involved networks.

5.1.2. Parameters and conditions

5.1.2.1. CELF. To obtain the accurate influence spread of a certain node set, we run Monte Carlo Simulations 10,000 times and compute the average number of activated nodes as the influence spread of the given node set in IC model, where the active probability is set to 0.01.

5.1.2.2. PageRank. We set the damping factor $\varepsilon = 0.85$, and suppose the convergence has been reached if the gap between a random node's PageRank score in the last iteration and the score in the current iteration is less than $g = 10^{-6}$.

5.1.2.3. SAEDV. We randomly initialize a node set and optimize it with iterations. The parameters for SAEDV are set as follows: the initial temperature $T_0 = 500{,}000$, $T_{f=}$ 100,000, the number of inner loop $q = 10$ and the temperature drop after each inner loop $\Delta T = 2000$.

5.1.2.4. DDSE. We give the descriptions of the parameters in Table 3. The best values for our algorithm will be determined one-by-one through the experiments in Section 5.2

5.1.2.5. Metrics. we use two conventional metrics, namely the influence spread and running time, to evaluate the accuracy and efficiency of algorithms, respectively. More specifically, the algorithm that achieves larger influence spread has better accuracy and the algorithm with less running time has higher efficiency.

5.1.2.6. Experimental condition. All experiments are performed on a Windows server with 2 CPUs (Intel Xeon E5-2660 V3, 2.6 GHz) and 128 GB memory. All the codes are written in Java.

5.2. Experiments for the parameters of DDSE

Before the comparison experiments, we conduct some experiments for the parameters of DDSE and find out the optimal values of these parameters. In our DDSE algorithm, we denote the size of population as n, the maximum number of iterations as g_{max}, the size of node set as K, the mutation probability as f, and the crossover probability as cr.

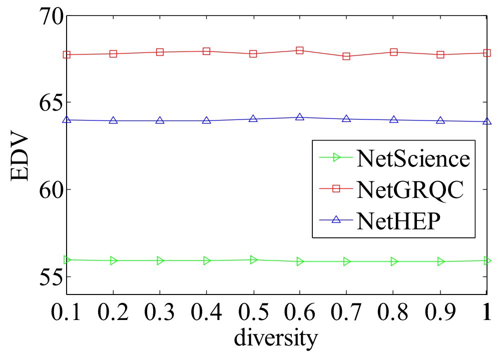


Fig. 6. The variations of fitness function EDV with different population diversity.

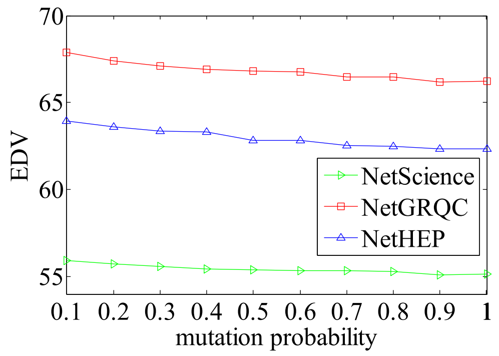


Fig. 7. The variations of fitness function EDV with different values of mutation probability.

5.2.1. Diversity of the population

DDSE employs the degree descending method to initialize the initial population, where the diversity of the population is controllable. To find out the best diversity, we run the algorithm on two data sets with the following fixed parameters: $n = 10$, $K = 50$, $g_{max} = 200$, $f = 0.1$, $cr = 0.5$ while the diversity varies between the range [0.1, 1] with the interval 0.1.

Fig. 6 shows the variations of fitness function EDV with different diversity values.

As shown in Fig. 6, with the increase of the diversity of population, the fitness values fluctuate slightly in these two social networks. In network NetGRQC and NetHEP, the fitness value is maximized when the diversity parameter is set to 0.6. While in network NetScience, the fitness of the population barely changes with the increase of the diversity. Therefore, we set the diversity of the population to 0.6.

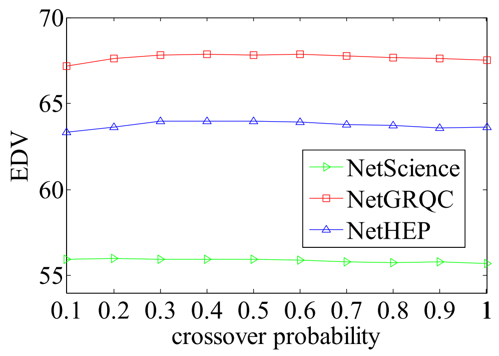


Fig. 8. The variations of fitness function EDV with different values of the crossover probability.

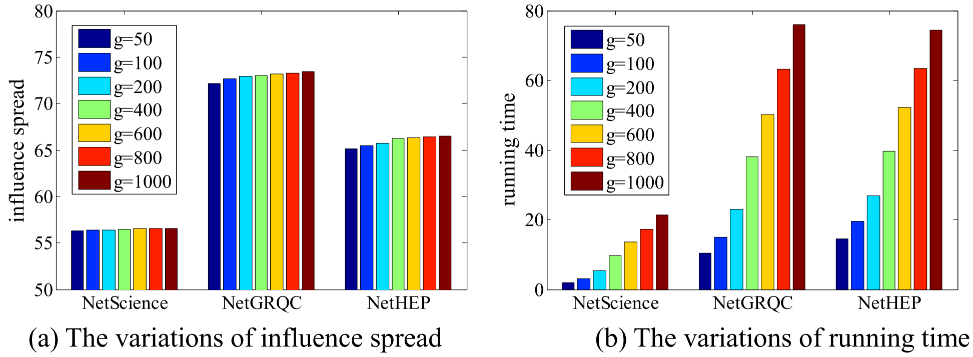


Fig. 9. The influence spread and time cost on three social networks with different number of iterations. (a) The variations of influence spread (b) The variations of running time.

5.2.2. Mutation probability

Mutations of population make it possible to find the optimal solution of influence maximization. The probability of the mutation reflects the speed of creating new solutions and it has a great impact on the convergence of the evolution. To explore the impact with different values of mutation probability, we set parameters: $n = 10$, $K = 50$, $g_{max} = 200$, $diversity = 0.6$, $cr = 0.5$. Fig. 7 shows the results with different values of the mutation probability.

As shown in Fig. 7, the fitness values reach the top with the mutation probability $f = 0.1$ and descend with the increase of the mutation probability for all networks. Therefore, we set the mutation probability to 0.1.

5.2.3. Crossover probability

To analyze the performance of DDSE with different values of the crossover probability, we set parameters: $n = 10$, $K = 50$, $g_{max} = 200$, $diversity = 0.6, f = 0.1$.

Fig. 8 shows the variations of the EDV values with different values of the crossover probability.

As shown in Fig. 8, the fitness value increases as the crossover probability increases in the interval [0.1, 0.4] but it decreases when the crossover probability increases in the interval [0.5, 1.0]. Therefore, we set the crossover probability cr to 0.4.

5.2.4. Maximum number of iterations

We compare the influence spread and running time while varying the maximum number of iterations with the fixed population size and other parameters. Fig. 9 shows the influence spread and running time in the social networks of NetGRQC and NetHEP, where $n = 10$, $K = 50$, $diversity = 0.6, f = 0.1$, $cr = 0.4$ and g denotes the maximum number of iterations.

As shown in Fig. 9, the influence spread increase slightly with the increase of the maximum iteration number. But compared with the dramatic increase of the running time caused by the increasing number of iterations, the gain of the influence spread is negligible. Therefore, we set the maximum number of iterations to 200 considering the trade-off between the performance and time cost.

5.2.5. Size of the population

We conduct experiments to compare the influence spread and running time under different population sizes, where other parameters are set as follows: the active probability $ap = 0.01$, $K = 50$, $g_{max} = 200$, $diversity = 0.6, f = 0.1$, $cr = 0.4$

As shown in Fig. 10, the influence spread after 200 iterations barely increases when the size of the population n increases from 5 to 100. Considering the trade-off between influence spread and running time, we set the size of population to 10.

5.3. Comparison experiments

We carry out extensive experiments and make comparisons between DDSE algorithm and other well-known algorithms for influence maximization in terms of influence spread to verify the good accuracy and running time to verify the high efficiency of our proposed DDSE algorithm.

5.3.1. Comparison for the influence spread (accuracy)

We run the algorithm with fixed active probability ($ap = 0.01$) and varying size of the seed set with the range from 5 to 50 on the networks of NetGRQC and NetHEP based on the IC model. Fig. 11 shows the experimental results of the five algorithms on these three networks, respectively.

As shown in Fig. 11, CELF and DDSE have similar performance on all three networks, exceeding all of other algorithms in term of influence spread. Regarding NetScience, all algorithms perform approximately because this network has a very low average degree. According to Fig. 11. (b), PageRank is close to SAEDV on network NetGRQC. Although PageRank and SAEDV reach the similar influence

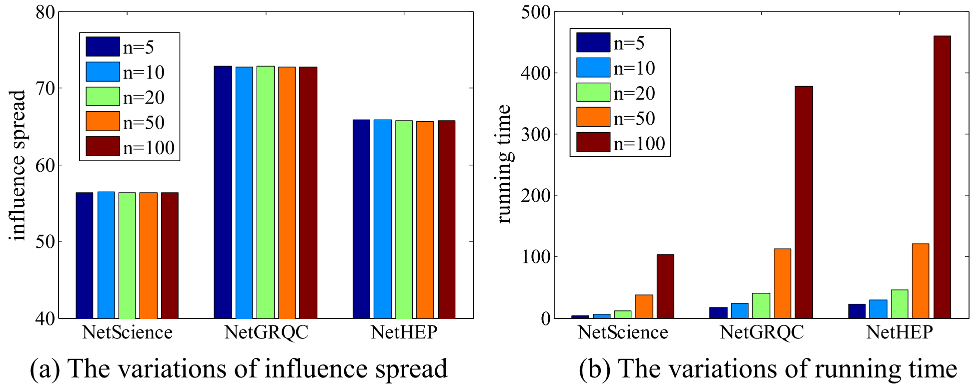
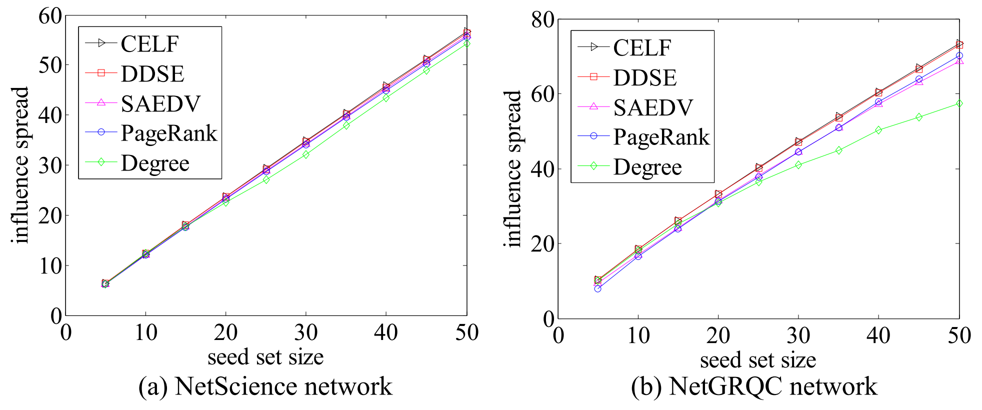


Fig. 10. The influence spread and time cost on three social networks with different population sizes, where active probability $ap = 0.01$, $K = 50$, $g_{max} = 200$, $diversity = 0.6, f = 0.1$, $cr = 0.4$. (a) The variations of influence spread (b) The variations of running time.



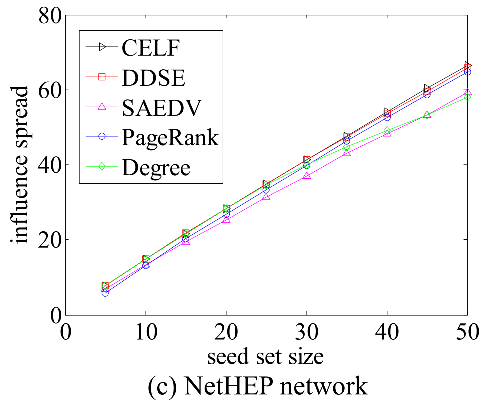
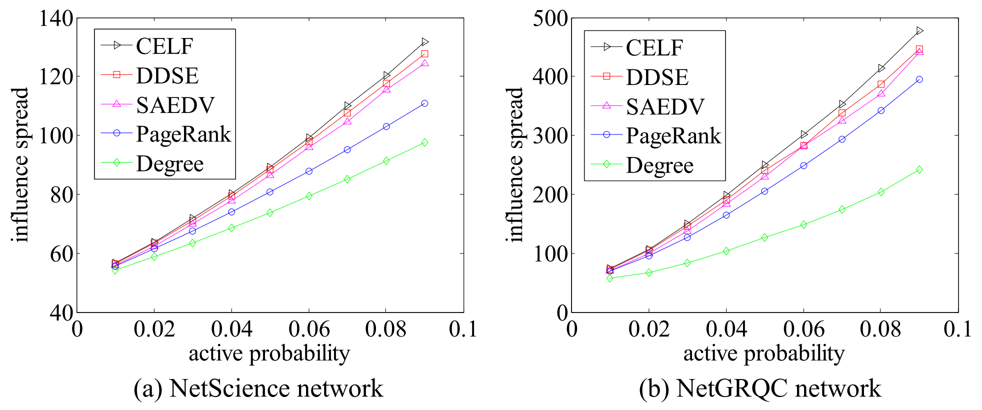


Fig. 11. The influence spread on three social networks based on IC model, where the active probability $ap = 0.01$ (a) NetScience network (b) NetGRQC network (c) NetHEP network.



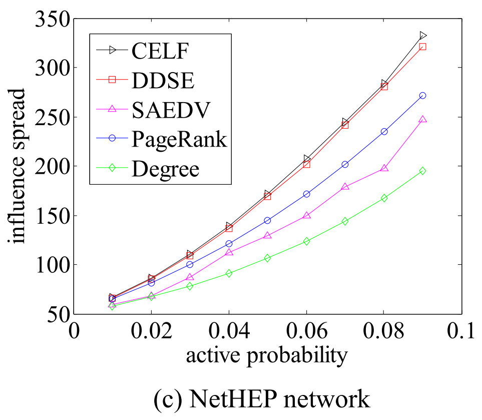


Fig. 12. The influence spread on three networks in the IC model with different values of the active probability, where $K = 50$. (a) NetScience network (b) NetGRQC network (c) NetHEP network.

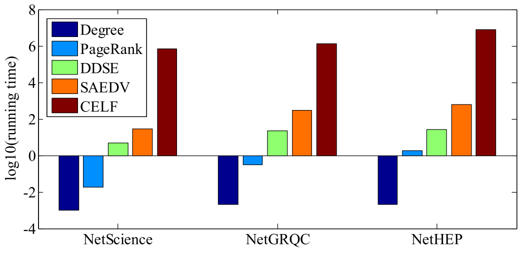


Fig. 13. The comparison of algorithms in time cost.

spread as DDSE when the seed set size is less than 20, they are inferior to DDSE when the seed set size is larger than 20. Especially for SAEDV, it is much worse than DDSE on network NetHEP due to its slow convergence. The degree centrality heuristic is the worst on all networks.

In the next step, we run all algorithms with different values of the active probability varying from 0.01 to 0.09 and the fixed seed set size ($K = 50$). Fig. 12 illustrates the results on three networks.

As shown in Fig. 12, with the increase of active probability, all algorithms reach a larger influence spread, although the increments are highly different. With the increase of active probability, DDSE keeps a close influence spread to CELF and expands its advantage over SAEDV, PageRank and degree centrality on all networks. In network NetGRQC, SEDV outperforms PageRank, but the situation is opposite on network NetHEP. This odd phenomenon can be explained as follows: NetHEP has a much larger scale than NetGRQC. The large scale exacerbates the disadvantage of SAEDV in efficiency due to its slow convergence characteristic. Degree heuristic has the worst performance on all networks. Both DDSE and SAEDV take the EDV as its objective function, but obviously DDSE works better than SAEDV in solution quality. The superiority of DDSE to SAEDV can be explained as follows: First, DDSE initialize the population with DDS-based method, which makes the population with much better quality because DDS fills the population with large-degree nodes in priority and large degree means more influential to some extent. Second, in the evolutionary process, DDSE creates new individuals with DDS-based method as well (which means nodes with large degree is selected into new solutions in a priority), rather than replacing the old individual with some random nodes. Again, our DDS strategy helps to improve the seed quality.

5.3.2. *Comparison for running time (efficiency)*

We keep records of the running time of each algorithm as well. Fig. 13 illustrates the comparison results of all involved algorithms in running time, where both the active probability ap and the seed set size k are fixed ($ap = 0.01$ and $K = 50$). In order to clear show the contrast between Degree heuristics and CELF, we take the logarithmic value of the running time instead of plotting the running time directly.

As shown in Fig. 13, the degree heuristics is the fastest approach, and it takes only 10^{-3} s to finish the computation. While CELF has the worst performance in running time due to the repeated Monte Carlo simulations in its procedures to find the most influential node. PageRank is also very fast and it ranks the second place. Replacing time-consuming repeated simulations with efficient estimation method and DDS-based operators, DDSE is five orders of magnitude faster than CELF. Compared with the SAEDV, DDSE is almost ten times faster. It can be explained as follows: DDSE employs the DDS-based evolutionary operators and the DDS strategy works efficiently in selecting the most influential nodes. Meanwhile because of the high efficiency of DDS, our evolutionary algorithm requires a very small-sized population. A small population means very little computational overhead.

6. Conclusion

In this paper, we tackle the influence maximization by proposing a novel evolutionary algorithm called DDSE. DDSE overcomes the efficiency issue of greedy approaches by eliminating repeated simulations with an estimation function EDV. We build the DDSE with our proposed degree-based heuristic called degree descending search and carefully designed operators: initialization, mutation, crossover and greedy selection. A neighborhood optimization called local search are utilized and operated on the seeds yielded during iterations to further improve it and achieve the final desirable solution. We conduct extensive experiments on the real-world social networks. Experimental results verify the effectiveness and incredibly high efficiency of our proposed algorithm, compared with the state-of-the-art greedy algorithm and other widely used approaches.

In the future, we also try to introduce some machine learning engines (such as Spark ML) to further improve the efficiency and effectiveness of our proposed algorithm

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 61402294, 61572328 and 61672358), the Major Fundamental Research Project in the Science and Technology Plan of Shenzhen (Grant nos. JCYJ20160310095523765, JCYJ20160307111232895, JCYJ20150630105452814).

References

Arab, M., Afsharchi, M., 2014. Community detection in social networks using hybrid merging of sub-communities. J. Netw. Comput. Appl. 40, 73–84.
Back, T., 1996. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press.
Bass, F.M., 1976. A New Product Growth Model for Consumer Durables, Mathematical Models in Marketing. Springer Berlin Heidelberg, 351–353.
Bond, R.M., Fariss, C.J., Jones, J.J., Kramer, A.D., Marlow, C., Settle, J.E., Fowler, J.H., 2012. A 61-million-person experiment in social influence and political mobilization. Nature 489 (7415), 295–298.
Brin, S., Page, L., 2012. Reprint of: the anatomy of a large-scale hypertextual web search engine. Comput. Netw. 56 (18), 3825–3833.
Brown, J.J., Reinegen, P.H., 1987. Social ties and word-of-mouth referral behavior. J. Consum. Res. 14 (3), 350–362.
Bryant, J., Miron, D., 2004. Theory and research in mass communication. J. Commun. 54 (4), 662–704.
Chen, W., Wang, Y., Yang, S., 2009. Efficient influence maximization in social networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 199–208.
Chen W., Wang C., Wang Y., 2010. Scalable Influence Maximization for Prevalent Viral Marketing in Large-scale Social Networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 1029–1038.
Chen W., Yuan Y., Zhang L., 2011. Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In: Proceedings of International Conference on Data Mining, IEEE, pp. 88–97.
Contractor, N.S., DeChurch, L.A., 2014. Integrating social networks and human social motives to achieve social influence at scale. Proc. Nat. Acad. Sci. USA 111, 13650–13657.
Cui, L., Yu, F.R., Yan, Q., 2016. When big data meets software-defined networking: sDN for big data and big data for SDN. IEEE Netw. 30 (1), 58–65.
Das, S., Suganthan, P.N., 2011. Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evolut. Comput. 15 (1), 4–31.
Domingos, P., Richardson, M., 2001. Mining the Network Value of Customers. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 57–66.
Goldenberg, J., Libai, B., 2001. Using Complex Systems Analysis to Advance Marketing Theory Development: Modeling Heterogeneity Effects on New Product Growth through Stochastic Cellular Automata 9. Academy of Marketing Science Review, 3.
Goldenberg, J., Libai, B., Muller, E., 2001. Talk of the network: a complex systems look at the underlying process of word-of-mouth. Mark. Lett. 12 (3), 211–223.
Gong, M., Yan, J., Shen, B., Ma, L., Cai, Q., 2016. Influence maximization in social networks based on discrete particle swarm optimization. Inf. Sci. 367, 600–614, (C).
Goyal, A., Lu, W., Lakshmanan, L.V.S., 2011. Celf++: optimizing the greedy algorithm for influence maximization in social networks, In: Proceedings of the 20th International Conference Companion on World Wide Web, ACM, pp. 47–48.
Granovetter, M., 1978. Threshold models of collective behavior. Am. J. Sociol. 83 (6), 1420–1443.
Han, M., Yan, M., Cai, Z., Li, Y., 2016. An exploration of broader influence maximization in timeliness networks with opportunistic selection. J. Netw. Comput. Appl. 63, 39–49.
Haralabopoulos, G., Anagnostopoulos, I., Zeadally, S., 2015. Lifespan and propagation of information in On-line Social Networks: a case study based on Reddit. J. Netw.

Comput. Appl. 56, 88–100, (C).
Heidari, M., Asadpour, M., Faili, H., 2015. SMG: fast scalable greedy algorithm for influence maximization in social networks. Phys. A: Stat. Mech. its Appl. 420, 124–133.
Jiang, Q., Song, G., Cong, G., Wang, Y., Si, W., Xie, K., 2011. Simulated annealing based influence maximization in social networks. In: Proceedings of AAAI Conference on Artificial Intelligence, AAAI Press, pp. 127–132.
Kempe, D., Kleinberg, J., Tardos, E., 2003. Maximizing the spread of influence through a social network. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 137–146.
Kimura, M., Saito K., 2006. Tractable models for information diffusion in social networks. Knowledge Discovery in Databases: PKDD 2006, pp. 259–271.
Lee, J.R., Chung, C.W., 2015. A query approach for influence maximization on specific users in social networks. IEEE Trans. Knowl. Data Eng. 27 (2), 340–353.
Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N., 2007. Cost-effective outbreak detection in networks, In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 420–429.
Lu, X., Yu, Z., Guo, B., Zhou, X., 2014. Predicting the content dissemination trends by repost behavior modeling in mobile social networks. J. Netw. Comput. Appl. 42 (3), 197–207.
Lv, Z., Song, H., Basanta-Val, P., Steed, A., Jo, M., 2017. Next-generation big data analytics: state of the art, challenges, and future research topics. IEEE Trans. Ind. Inform. 13 (4), 1891–1899.
Mahajan, V., Muller, E., Bass, F.M., 1990. New product diffusion models in marketing: a review and directions for research. J. Marketing 54 (1), 1–26.
Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., et al., 2016. Mllib: machine learning in apache spark. J. Mach. Learn. Res. 17 (1), 1235–1241.
Newman, M.E., 2006. Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 74 (3 Pt 2), 036104.
Onnela, J.P., Reed-Tsochas, F., 2010. Spontaneous emergence of social influence in online systems. Proc. Nat. Acad. Sci. USA 107, 18375–18380.
Peng, S., Yu, S., Yang, A., 2014. Smartphone malware and its propagation modeling: a survey. IEEE Commun. Surv. Tutor. 16 (2), 925–941.
Quick, D., Choo, K.K.R., 2017. Pervasive social networking forensics: intelligence and evidence from mobile device extracts. J. Netw. Comput. Appl. 86, 24–33.
Rahimkhani, K., Aleahmad, A., Rahgozar, M., Moeini, A., 2015. A fast algorithm for finding most influential people based on the linear threshold model. Expert Syst. Appl. 42 (3), 1353–1361.
Richardson, M., Domingos, P., 2002. Mining knowledge-sharing sites for viral marketing. In: Proceedings of the Eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 61–70.
Sun, J., Ma, H., 2014. Heterogeneous-belief based incentive schemes for crowd sensing in mobile social networks. J. Netw. Comput. Appl. 42, 189–196.
Wang, Y., Cong, G., Song, G., Xie, K., 2010. 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, pp. 1039–1048.
Wang, Y., Yin, G., Cai, Z., Dong, Y., Dong, H., 2015. A trust-based probabilistic recommendation model for social networks. J. Netw. Comput. Appl. 55, 59–67.
Wen, S., Jiang, J., Liu, B., Xiang, Y., Zhou, W., 2017. Using epidemic betweenness to measure the influence of users in complex networks. J. Netw. Comput. Appl. 78, 288–299.
Wu, H., Wang, X., Peng, Z., Li, Q., 2013. Div-clustering: exploring active users for social collaborative recommendation. J. Netw. Comput. Appl. 36 (6), 1642–1650.
Zhang, Y., Gu, Q., Zheng, J., Chen, D., 2010. Estimate on expectation for influence maximization in social networks. In: Proceedings of Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. Springer-Verlag, pp:99-106.
Zhang, Y., Song, B., Zhang, P., 2017. Social behavior study under pervasive social networking based on decentralized deep reinforcement learning. J. Netw. Comput. Appl. 86, 72–81.
Zhou, Y., Zhang, B., Sun, X., Zheng, Q., Liu, T., 2017. Analyzing and modeling dynamics of information diffusion in microblogging social network. J. Netw. Comput. Appl. 86, 92–102.

Laizhong Cui received the B.S. degree from Jilin University, Changchun, China, in 2007 and Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012. He is currently an associate professor in the College of Computer Science and Software Engineering at Shenzhen University, China. He led a project of the National Natural Science Foundation, and several projects of Guangdong Province and Shenzhen City. His research interests include recommendation system, content distribution, computational intelligence, software-defined network and social network.

Huaixiong Hu is working toward the M.Sc. degree at College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include data mining, social network, and multi-objective optimization.

Shui Yu is currently a Senior Lecturer of School of Information Technology, Deakin University. He is a member of Deakin University Academic Board (2015–2016), a Senior Member of IEEE, and a member of AAAS and ACM, the Vice Chair of Technical Committee on Big Data Processing, Analytics, and Networking of IEEE Communication Society, and a member of IEEE Big Data Standardization Committee. Dr Yu's research interest includes Security and Privacy in Networking, Big Data, and Cyberspace, and mathematical modelling. He has published two monographs and edited two books, more than 150 technical papers, including top journals and top conferences, such as IEEE TPDS, IEEE TC, IEEE TIFS, IEEE TMC, IEEE TKDE, IEEE TETC, and IEEE INFOCOM. Dr Yu initiated the research field of networking for big data in 2013. His h-index is 25.

Yan Qiao is a professor at the College of Computer Science and Software Engineering at Shenzhen University. She received her Ph.D. degree in information and communication engineering from Xidian University, Xioan, China, in 2003. From 2013–2014 she worked at Carleton University, Ottawa, Canada, as a visiting scholar. Her research interests are in network security, cloud computing, and software-defned networking. Her current focus is research and development of security in software defned networking.

Zhong Ming is a Professor with the College of Computer and Software Engineering, Shenzhen University. He is a Senior Member of the Chinese Computer Federation. He led three projects of the National Natural Science Foundation, and two projects of the Natural Science Foundation of Guangdong, China. His major research interests include home networks, Internet of Things, and cloud computing.

Zhenkun Wen received the M.E. degree in computer science and engineering from Tsinghua University, Beijing, China, in1999. He is currently a Professor in College of Computer Science and Software Engineering at Shenzhen University, Guangdong, China. His research interests include computer networks, machine learning, video information security,etc.

Nan Lu is a professor in the College of Computer Science and Software Engineering at Shenzhen University, China. He received the Ph.D. degree from Jilin University, Changchun, China. He led several projects of Guangdong Province and Shenzhen City. His research interests include commerce intelligence, machine learning, complex network community structure, and trust mining of social network.