Physica A 478 (2017) 20–30

Contents lists available at ScienceDirect

Physica A

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

Maximizing influence in a social network: Improved results using a genetic algorithm

Kaiqi Zhang[a,b], Haifeng Du[b], Marcus W. Feldman[b,c,*]

[a] *School of Management of Xi'an Jiaotong University, Xi'an, Shanxi Province, 710049, China*
[b] *Center for Administration and Complexity Science of Xi'an Jiaotong University, Xi'an, Shanxi Province, 710049, China*
[c] *Morrison Institute for Population and Resource Studies, Stanford University, Stanford, CA 94305, United States*

HIGHLIGHTS

- The genetic algorithm approach gives optimal results.
- Diversity of solutions is maintained.
- The genetic algorithm performs better than most other algorithms.

ARTICLE INFO

Article history:
Received 30 September 2016
Received in revised form 13 December 2016
Available online 28 February 2017

Keywords:
Influence maximization problem
Social network
Simulated annealing
Genetic algorithm

ABSTRACT

The influence maximization problem focuses on finding a small subset of nodes in a social network that maximizes the spread of influence. While the greedy algorithm and some improvements to it have been applied to solve this problem, the long solution time remains a problem. Stochastic optimization algorithms, such as simulated annealing, are other choices for solving this problem, but they often become trapped in local optima. We propose a genetic algorithm to solve the influence maximization problem. Through multi-population competition, using this algorithm we achieve an optimal result while maintaining diversity of the solution. We tested our method with actual networks, and our genetic algorithm performed slightly worse than the greedy algorithm but better than other algorithms.

1. Introduction

A social network reflects the relationships among individuals that allow information to spread. Via social networks, person-to-person communication of information, ideas, and behaviors can occur. Compared to the traditional information-diffusion model (such as the SIS/SIR epidemic model) [1–4], the network approach provides a more realistic platform for the spread of information, including knowledge innovation and marketing (also known as "word of mouth" or "viral marketing" [5,6]).

The influence maximization (denoted as IM) problem entails finding a small subset of a network's nodes, called the "seed set", which can influence the largest number of nodes via a model of spread in the network [7–9]. There are two representative diffusion models: the linear threshold model (denoted as LT) and the independent cascade model (denoted as IC) [9]. Kempe et al. demonstrate that the IM problem under both LT and IC is an NP-hard problem, and propose a basic greedy algorithm (denoted as KGA) to solve it [9]. Chen et al. propose another greedy algorithm (denoted as CGA) that improves the

* Corresponding author at: Morrison Institute for Population and Resource Studies, Stanford University, Stanford, CA 94305, United States.
E-mail address: mfeldman@stanford.edu (M.W. Feldman).

http://dx.doi.org/10.1016/j.physa.2017.02.067
0378-4371/© 2017 Elsevier B.V. All rights reserved.

KGA by pruning the graph [10], while the cost-effective lazy forward (denoted as CELF) method, due to Leskovec et al. [11], employs local search to address the IM problem. Although the CELF method is still a greedy algorithm, it is 700 times faster than the KGA and thus is easier to apply to some networks [11]. However, since the IM problem is NP-hard, due to the necessity of repeatedly calculating the influence of every node, all of these greedy algorithms still take a great deal of time to achieve results.

Some heuristics have been proposed to improve the efficiency of the greedy algorithms. Kempe et al. propose three main heuristics–random, degree, and centrality for their algorithm [9]. Based on the degree heuristic, Chen et al. propose the DegreeDiscount heuristic for the IC [10]. Although they save runtime, heuristics also reduce the accuracy of the results obtained with these algorithms.

The simulated annealing algorithm proposed by Jiang [12] (denoted as SA) is a stochastic optimization algorithm used to solve the IM problem. The optimization mechanism in SA replaces the diffusion simulations in greedy algorithms such as KGA, CELF, and CGA. It reduces the computation cost and decreases the runtime. Distant Node heuristic and Single Node Spreading heuristic are two heuristics applied with the SA to find the neighborhood of a solution by local searching. The results of SA outperform the results of random, degree, DegreeDiscount heuristic and CGA. Also the SA is faster than the CGA by two orders of magnitude with the heuristics. But due to the limitation of the simulated annealing mechanism, SA finds it hard to obtain multiple optimal solutions, especially in the LT model. Also the results of SA may be affected by the parameters.

In this paper, we target the LT with a fixed choice of thresholds, which is rarely mentioned in previous studies. We propose a genetic algorithm for the IM problem (denoted as GA). Specifically, we set the influence spread of the seed set as the fitness function, and different seed set populations achieve optimal solutions through competition and evolution. Unlike simulated annealing, the setting of multi-populations in GA can guarantee the diversity of algorithm solutions when the threshold in LT is fixed. In addition, we modify some heuristics in SA to improve the GA's accuracy and efficiency.

In Section 2 related studies are reviewed, including greedy algorithms, their heuristics and SA. In Section 3 we present our genetic algorithm, which we test in Section 4 by solving the IM problem for four actual networks, and compare our GA to other algorithms. Section 5 summarizes our conclusions.

2. Related work

2.1. The influence maximization problem and greedy algorithms

The IM problem was initially described as an optimization problem by Kempe. The objective of the IM problem is to find a seed set A of k nodes such that the number of influence nodes from seed set A is as large as possible. The influence process of a seed set unfolds under a diffusion model (LT or IC). Although the mechanisms in LT and IC are not the same, both follow the same process: as time unfolds, more and more neighbor nodes of an inactive node become active; at some point one or more neighbor nodes may influence or activate the original node, and this inactive node, when active, may in turn trigger further switches by nodes to which it is connected [9].

Kempe proved the IM problem to be a NP-hard problem caused by submodularity, which produces a decrease in the increment of activated nodes at each iteration. KGA was first regarded as an efficient approximation method for solving IM problem [9]. With the KGA algorithm we can at least obtain a $(1–1/e)$ approximate result; this result is considered to be an optimization criterion. However, because KGA is a hill-climbing greedy algorithm, the nodes in the network need to be traversed repeatedly at every iteration, and each node needs R rounds of simulation (generally $R = 20{,}000$) to evaluate the contribution, the efficiency of the algorithm is reduced.

To improve the efficiency of KGA, a number of algorithms or methods have been proposed. The CELF method proposed by Leskovec et al. is about 700 times faster than KGA by utilizing the submodularity in the influence process [11]. CGA proposed by Chen et al. [10] is another algorithm based on the greedy algorithm, and its optimization mechanism traces to the certification process in Kempe et al. [9]. CGA attempts to acquire a new graph based on the influence spread and to find the seed set from the new graph. Other improved algorithms include the community-based algorithm [13] and a modified CELF++ method [14], and their efficiency seems significantly higher than that of KGA. A common theme in the above algorithms (including KGA) is to find a seed set containing nodes that may have the optimal average influence in the network. Each node needs several rounds of simulation to evaluate its influence. Therefore for large-scale networks, the algorithms above still take a long time to obtain the result.

2.2. Heuristics in greedy algorithms

Some heuristics have been proposed as time-saving solutions of the IM problem. Kempe et al. demonstrated that there are three simple heuristics: random, degree, and centrality [9]. The random heuristic selects nodes randomly, without considering node influence, to form the seed set in the network. Degree and centrality heuristics derive from the definition of the node influence in social networks [15], and while degree and centrality heuristics are more accurate than the random heuristic, they lead to less accurate results in KGA and do not give significantly better solutions of the IM problem.

Based on the degree heuristic, another heuristic, DegreeDiscount, targets the IM problem in CGA by taking into account prior knowledge of the node's neighbors [16]. It assumes that if node u has been selected in the seed set, then, when selecting its neighbor node v as a new seed based on the degree, the edge $e(u, v)$ should not be counted toward the degree of v. That is, prior knowledge is used for the node selection and greatly improves the algorithm's accuracy. Nevertheless, DegreeDiscount fails to precisely estimate the influence spread of nodes and its accuracy may still be improved.

Other heuristics have been used to solve the IM problem, such as Logical Directed Acyclic Graph [17], Simpath [18], and so on, which were designed to improve the algorithmic efficiency using some prior knowledge about nodes, such as degree, centrality, or neighbor nodes. However, these heuristics seldom evaluate the global influence of a node or they cost a great deal of time accessing and storing prior knowledge. Also, they do nothing about changing the local search region and may lead to solutions that are trapped in local optima. Therefore, multi-neighborhood solutions are needed for problems as complicated as the IM problem.

2.3. Stochastic optimization algorithm and SA

The stochastic algorithm has higher efficiency and accuracy than the greedy algorithm for solving some optimization problems. The SA is a typical stochastic optimization algorithm for the IM problem and outperforms CGA and the DegreeDiscount heuristic [12]. Its main structure is shown below as Algorithm 1.

```
Algorithm 1 Simulated Annealing algorithm
Input: Graph $G = (V, E)$, Seed set scale $K$, Initial temperature $T_0$, Termination temperature $T_f$, the Number of inner
    loop $q$, The amount to cut down the current temperature in the outer loop $\Delta T$
Output: Seed Set $A$
1: $t \leftarrow 0, T_t \leftarrow T_0, count \leftarrow 0$;
2: Select an initial seed set $A \subset V, |A| = k$, randomly ;
3: while $T_t < T_f$ do
4:   calculate $\sigma(A)$
5:   $A' \leftarrow F(A, G)$ ;
     {Create a neighbor solution set:DNS&SNSH}
6:   $count++$
7:   calculate the change of the fitness $\Delta f \leftarrow \sigma(A') - \sigma(A)$
8:   if $\Delta f > 0$ then
9:     $A \leftarrow A'$;
10:  else
11:    create a random number $\xi$;
12:    if $\exp(\frac{\Delta f}{T_t}) > \xi$ then
13:      $A \leftarrow A'$;
14:    end if
15:  end if
16:  if $count > q$ then
17:    $T_t \leftarrow T_t - \Delta T, t \leftarrow t + 1, count \leftarrow 0$;
18:  end if
19: end while
20: return $A$;
```

SA generates a k-node seed set initially and uses simulated annealing in searching for the optimal solution with parameters initial temperature T_0, termination temperature T_f, and the amount by which to reduce the current temperature ΔT. SA estimates the influence spread of a seed set A containing k-nodes, and because there is no need to add nodes into the seed set at every iteration, the runtimes of the SA will not be affected by the size of network, but only by the parameters. Therefore the convergence of SA may be affected by the parameters, which tend to cause SA to be trapped in local optima.

Two heuristics, the Distant Node Heuristic (denoted as DNH) and the Single Node Spreading Heuristic (denoted as SNSH), in SA aim to increase efficiency and accuracy. Nonetheless, the parameters of heuristics above may also increase the length of the search path and affect the convergence rate or the final result of the algorithm. Thus the heuristics in SA are worth using for improving the efficiency and accuracy of the algorithm we propose.

3. Proposed algorithm

3.1. Problem statement

Table 1 gives important symbols used in this paper. For a network G, the nodes in V can be set to "active" or "inactive". We call an influence process from one active node or a set of active nodes "influence spread". As influence spreads under a diffusion model (LT or IC), the nodes in network can switch from being inactive to active. We denote $\sigma(v)$ as the set of influence nodes (switching from being inactive to being active) from any node v. The IM problem tries to find a set of k-nodes A such that $\sigma(A)$ is maximized according to one diffusion model.

We adopt LT as a diffusion model in this paper. In Kempe et al.'s work [9], to prove that the influence spread of LT is submodular, they randomly selected the thresholds of nodes and treated LT as a stochastic model. In KGA [9], the threshold of each node is re-chosen randomly in each simulation to compute and evaluate the average influence for each node. Thus in KGA and its subsequent greedy algorithms (including CELF, CGA), $\sigma(A)$ is an average maximized set via R rounds of simulation.

Table 1
Symbols.

Notations	Descriptions
$G = (V, E)$	A network with nodes set V and edges set E
V	Nodes set
E	Edges set
N	Number of nodes in G, or the size of V
M	Number of edges in G, or the size of E
A	Seed set, $A = \{v_1, v_2, \ldots, v_k\}$
k	The size of seed set A
$\sigma(v)$	The set of nodes that the node v can influence
R	The number of the rounds of simulation in greedy algorithms

However, different thresholds randomly selected in each round of simulation may give different influence spread. Whether the influence of one node under different influence spread in each round of simulation can be accumulated and averaged remains questionable. In addition, for a fixed choice of thresholds, there may be several different seed sets that give the same influence results (i.e. the same number of influence nodes). However, the above greedy algorithms may record only one solution, and rarely discuss the diversity of solutions. Although the optimization mechanism of SA is different from that of greedy algorithms, SA does not address the diversity of solutions. Because the SA was originally designed for IC, and the optimization mechanism of SA is single-point search, SA finds it hard to obtain more than one optimal solution for a fixed choice of thresholds in LT. Thus in this paper, in order to discuss the diversity of solutions for a fixed choice of thresholds in LT, we introduce a multi-population heuristic algorithm–a genetic algorithm–for solving the IM problem, and to guarantee the diversity of the solution.

The genetic algorithm was first proposed by John Holland in 1975 [19]. It is a metaheuristic inspired by the process of natural selection. In particular, using a multi-population stochastic algorithm; competition among and evolution of the populations is conducive to optimization. The multi-population setting in the genetic algorithm ensures the diversity of the solution. Thus a genetic algorithm is commonly used to generate diverse solutions to optimization problems. As a combinatorial optimization problem, the IM problem must be solved under multiple constraints (seed set size, diffusion model, network structure, etc.). Therefore a genetic algorithm has certain advantages for solving the IM problem and searching for diverse solutions when thresholds in LT are fixed.

Compared to SA, the GA has three advantages for solving the IM problem under LT with a choice of thresholds. First GA initially generates multiple k-node seed sets (called "populations" in this paper). Based on the fitness function, an optimal population can be selected at each iteration. Thus the GA is a multi-point local search process. We set a clone operation in GA, and with this operation, we can expand the range in local search while ensuring the diversity of solutions. Thus we can guarantee the diversity of the solution in LT for a fixed choice of thresholds. Second we modify some heuristics in SA to improve the efficiency and accuracy of GA. Finally, we modify the application of prior knowledge by using a new heuristic that can adjust the prior knowledge about each node at each iteration.

3.2. Main algorithm framework

The main structure of GA is shown as Algorithm 2 below.

```
Algorithm 2 Main structure of proposed Genetic Algorithm
Input: Graph $G = (V, E)$, Threshold $\theta$, Seed set scale $k$, Population number $m$, Clone population number $n$, Crossover
    probability $p_c$,Mutation probability $p_m$;
Output: Set of seed set $A_i$ :$T\{A_i\}$
1:  Calculate influence of each node $v$:$\sigma(\{v\})$ as prior knowledge $L$ ;
2:  Initialization population group  and population $\tau_i \in T, \tau_i = \{v_1, v_2 \ldots v_k\}$ $(i = 1 \ldots m)$;
3:  while Termination Condition do
4:      Clone generate $n$ clone-population group $T_i$: ${\tau_i}^l \leftarrow \tau_i$ $(l = 1 \ldots n)$;
5:      if random $\xi_c > p_c$ then
6:          random select two populations ${\tau_i}^a, {\tau_j}^b$
7:          Crossover Operator:${\tau_i}^a \leftrightarrow {\tau_j}^b$;
8:      end if
9:      for each population do
10:         if random $\xi_m > p_m$ then
11:             Mutation Operator: ${\tau_i}^{l'} \leftarrow {\tau_i}^l$;
12:         end if
13:     end for
14:     Selection: calculate each populations $\sigma({\tau_i}^l)$
        select $max|\sigma({\tau_i}^l)|$ in each clone-populations $T_i$ :$\tau_i = \arg\max(\left|\sigma\left(\tau_i^l\right)\right|)$;
15:     Generate new population group $T$;
16: end while
17: return $T\{A_i\}$;
```

In the GA, we need to calculate the influence spread of nodes initially as the prior knowledge for each node. The prior knowledge contains the activated nodes or influence nodes of each node v and the distances between these activated nodes and each node v. Initialization creates the population for the genetic algorithm. Let population $\tau_i = \{v_1, v_2, \ldots, v_k\}$ $(i = 1, 2, \ldots, m)$ denote one seed set, where k is the scale of the seed set. We create m populations that initially denote the seed set group T. In order to highlight the randomness in GA, the selection of nodes at initialization is random. In the clone operation (line 4), each population τ_i will copy itself to generate clone populations. $\tau_i^l = \{v_1, v_2, \ldots, v_k\}$ $(i = 1, 2, \ldots, m;$ $l = 1, 2, \ldots, n)$, which means for each τ_i, there are n clone populations τ_i^l with the same nodes as in τ_i. The genetic operator (lines 5–13) includes a two-operator–crossover operation (line 7) and a mutation operator (line 11). We use three heuristics with these two operators to achieve the local search. In the selection operation (line 14), we use the influence number of the seed set as our fitness function. We select the optimal seed set from each clone population group to obtain at least m optimal solutions. Thus we can obtain multiple different seed sets. The termination condition of the GA is the influence number of an optimal seed set that remains at a steady state.

We design a data storage structure to store the populations in GA by setting each population as an independent entity to be stored in computer memory. Each node in the population is independent; two nodes in different populations, even if they have the same ID or label in graph, are stored separately. For example, node v may be one of the elements in both populations τ_i and τ_j, but a change in properties of node v in population τ_i does not affect properties of node v in τ_j stored in computer memory. Therefore we can apply heuristics to the population to fully guarantee the independence of the populations. We give details of the application below.

3.3. The heuristics for local search

There are three heuristics in GA, which form the crossover and mutation operators. They are elite cooperation heuristic (ECH), influence distance heuristic (IDH), and elitism heuristic (EH). ECH is for crossover, and IDH and EH are for mutation. As basic information for these heuristics, we introduce prior knowledge about nodes. For the CELF algorithm, for instance, we need to estimate each single node's influence Δs_v and its activated nodes initially, then store both of them using the data storage structure described above.

I: Crossover operator

In GA, crossover is a node exchange process between populations. Here we construct an elite cooperation heuristic (ECH) to achieve the crossover operator. Specifically, we assume a node with a high Δs_v is an elite node. We randomly select two populations, τ_i and τ_j, from all populations. Then we sort the nodes in population τ_i by descending order of Δs_v and sort the nodes in population τ_j by ascending order. Here we use the single-point crossover.

As shown in Fig. 1, we randomly select a flag position from the left in population τ_i, and all nodes beyond that position are swapped between the two populations. In this way, the elite nodes will be included in one population and the sub-elite nodes will be placed in another population. Although we cannot guarantee that a node's Δs_v in a new combination is still optimum, we can at least remove the non-optimum nodes in former combination. Such a setting should be conducive to the search for an optimal solution. The specific algorithm is shown in Algorithm 3.

```
Algorithm 3 Crossover Operation:Elite Cooperation Heuristic
Input: Crossover populations$\tau_i$ and $\tau_j$
Output: New populations $\tau_i'$ and $\tau_j'$
 1: Sort $\tau_i$ by descending $\Delta s_v$: $\tau_i' \leftarrow \tau_i$
 2: Sort $\tau_j$ by ascending $\Delta s_v$: $\tau_j' \leftarrow \tau_j$
 3: Random flag $f$ , $f < k$;
 4: for $n = 1 : f$ do
 5:     $v_n \in \tau_i, u_n \in \tau_j$;
 6:     if $v_n \notin \tau_j || u_n \notin \tau_i$ then
 7:         $v_n \leftrightarrow u_n$;
 8:     end if
 9: end for
10: return $\tau_i'$ and $\tau_j'$;
```

In GA, the populations in the crossover operation are from the different clone-population groups. We can enlarge the search space and possibly obtain an optimal solution beyond a local search.

II: Mutation operator

We propose two heuristics –the influence distance heuristic (IDH) and the elitism heuristic (EH) in the mutation operator. The IDH is similar to the DNH in SA. DNH assumes that the nodes within a short distance may share the same range of influence so they should be disregarded in the process of finding a neighborhood solution. The disregarded set $M(A, d)$ satisfies

$$M(A, d) = \{u | u \in V - A, \forall v \in A, dis(u, v) \leq d\}, \quad (1)$$

where $dis(u, v)$ is the length of the shortest path from node u to node v, and d is a parameter. The difference lies in the fact that the distance in DNH is a topological distance, while the distance in GA is based on the influence spread of nodes.

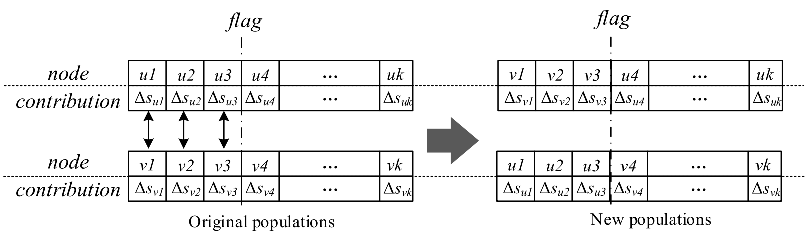


Fig. 1. The crossover operator. The figure on the left shows two original populations while the one on the right shows two new populations after the crossover. We swap the nodes and their corresponding contributions on the left side of the flag in the original populations, and the new populations contain both the original nodes and the exchanged nodes with their contributions.

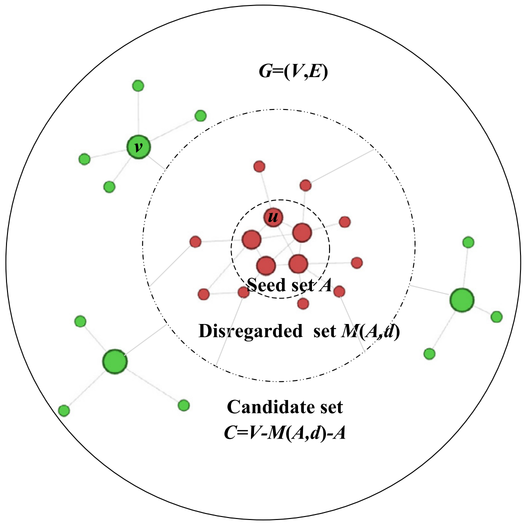


Fig. 2. Illustration of the node set V of network $G(V, E)$ and its three components: candidate set C, disregarded set $M(A, d)$, and seed set A in the mutation operator. The network structure represents the set of nodes V. The path length between two nodes is the influence distance. The max-distance between the nodes in A and the nodes in $M(A, d)$ is d, and the min-distance between the nodes in C and nodes in A is larger than d. Node u in A can only activate one node, and node v in C can activate four nodes. Node u and node v will be swapped in the mutation operation.

Specifically, since we initially have the influence node of every node, set D_v refers to the furthest influence distance of node v. We find a set $M(A, d)$ and a candidate set of nodes $C = V$-$M(A, d)$-A (as shown in Fig. 2), and as in DNH, disregard the nodes in $M(A, d)$ and select the nodes in C when creating the neighbor seed set. The specific algorithm is shown in Algorithm 4.

```
Algorithm 4 Mutation operator:Influence Distance Heuristic
Input: Seed set $A$; Graph $G = (V, E)$;Node prior knowledge $L$
Output: Candidate node set $C$;
1: $M \leftarrow \emptyset$
2: for each node $v$ in $L$ do
3:     $D_v \leftarrow$node $v$'s influence distance;
4: end for
5: $D \leftarrow D_v$;
6: Set $d$: $2 \le d \le D$;
7: for each node $v$ in seed set $A$ do
8:     $M_v \leftarrow$ node $v$ influence nodes set
9:     add $M_v$ to $M$;
10: end for
11: $C \leftarrow V - M - A$;
12: return $C$;
```

In order to avoid set C being empty, we give d some bounds: (1) d should be less than the largest influence distance D_v and larger than or equal to two; (2) d should be adjusted by the seed set scale k and the size of the network. However, as the network structure varies, we are unable to give a precise formula for the value of d. Compared to DNH, our heuristics add more prior knowledge. The nodes in set C are more different than the nodes in the seed set A, and the bounds on d may decrease the likelihood that set C is empty.

Table 2
Experimental networks tested.

Network	Nodes	Edges	Average degree	Density
NETSCIENCE	1589	2,742	3.451	0.002
EMAIL	167	5,784	34.635	0.209
INFECTIOUS	410	5,530	13.488	0.033
UCSOCIAL	1899	20,296	10.688	0.006

EH is another heuristic in mutation operator, and is shown in Algorithm 5.

```
Algorithm 5 Mutation operator:Elitism Heuristic
Input: Seed set $A$; Graph $G = (V, E)$;
Output: Mutation Seed Set $A'$
1: candidate node set $C \leftarrow IDH(A, G)$;
2: Random set $\lambda \leq size(A)$ and $\lambda \leq size(C)$
3: Exchange set $E_A \leftarrow roulette(A)$,$E_C \leftarrow roulette(C)$
4: Set $A' \leftarrow A$
5: for each node $v$ in $E_A$ and $u$ in $E_C$ do
6:     remove node $v$ from $A'$ and add node $u$ into $A$;
7: end for
8: return $A'$;
```

EH not only complies with the essence of the genetic algorithm, but also preserves the most influential nodes in the seed set A, while adding possibly influential nodes from C. Combined with IDH, the nodes in candidate set C may have a different range of influence spread from those in seed set A. As shown in Fig. 2, we assume the nodes influenced by candidate node set C may be relatively independent of seed set A. We can guarantee an optimal solution in local search by removing a node u with a lower Δs_v from A and adding a node v in C with a higher Δs_v to A.

Based on the heuristics above, we obtain a new population from the original population via crossover and mutation. Using the fitness function $\sigma(\tau_i)$, we select an optimal population from every clone-population group as the next iteration's initial population. Note that when calculating $\sigma(\tau_i)$, we add the nodes in τ_i stepwise to estimate Δs_v of each node. This serves as prior knowledge in the next iteration. In a specific implementation, we trigger the crossover operator with probability p_c and the mutation operator with probability p_m.

Our algorithm does not improve the computational efficiency of calculating the influence spread $\sigma(A)$ of seed set A. Because we have initially fixed the number of nodes in the seed set, the algorithm does not need to calculate the influence spread for every node (as some greedy algorithms do) at each iteration. Thus the runtime of each population in each iteration to calculate the influence spread is $O(kL)$, where k is the number in the seed set, and L is the runtime for calculating $\sigma(A)$. The computation cost of the algorithm is $O(tmnkL)$, where m represents the number in the population, t represents the iteration number, and n represents the number of clone populations. Although, formally, the runtime used by GA is equal to the KGA's $O(kVRLM)$ and SA's $O(tRML)$ (t is the number of iterations, M is the number of edges), in practice $tmn << VRM$ and $mn << RM$. Thus the GA's computation time should be less than that of KGA and SA.

Overall, the GA we propose is designed as a classic genetic algorithm. Therefore the convergence of GA is consistent with that of classic genetic algorithms.

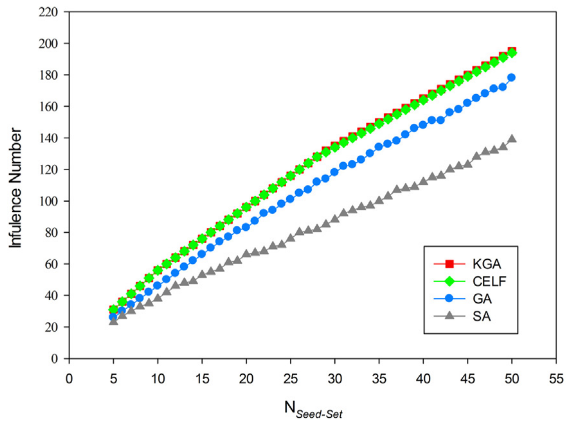
4. Experiments

4.1. Experimental setup

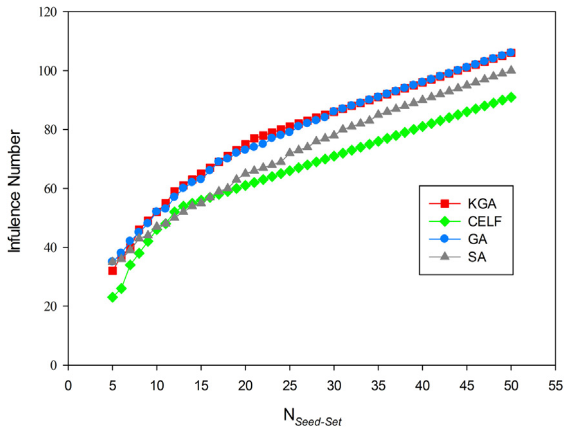
Two experiments test our GA and the results are compared with KGA, CELF, and SA for the IM problem. First, we try to compare the solutions using different algorithms with different seed set sizes, and compare the number of solutions under a certain spread scenario. Second, the experiment tests the genetic algorithm with different parameters.

The first experiment is based on four actual networks. The characteristics of these four networks are shown in Table 2. The degrees of nodes and the number of edges will affect the efficiency of algorithm; we list average degree and density in Table 2 as characteristics of network. In this experiment, we set the threshold in the LT diffusion model subject to the uniform [0,1] distribution. "NETSCIENCE" [20] is a co-authorship network of scientists working on network theory. "EMAIL" is an internal email communication network between employees of a mid-sized manufacturing company [21], and this network is merged by a dynamic network, so the edge weights represent the number of mails between two nodes in a certain period. "INFECTIOUS" is a network of visitors in the exhibition INFECTIOUS: STAY AWAY [22]. Edges represent face-to-face contacts between the visitors. This network is also a dynamic network; in our test we merge the edges and make their weights signify the number of contacts. "UCSOCIAL" is a message network between the users of an online community of students from the University of California, Irvine [23], where the edge weights represent the number of messages.

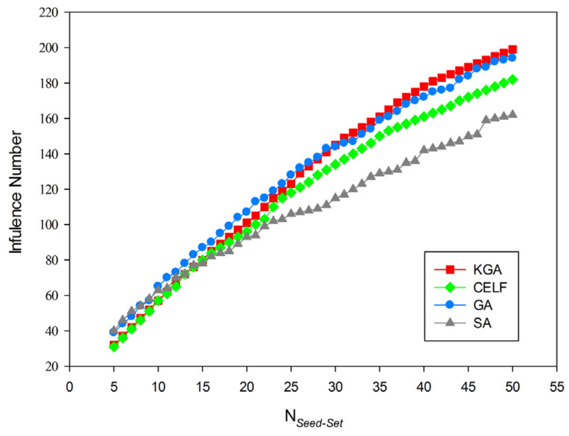
In the first experiment, all the influence spread is under a fixed choice of thresholds. Therefore, the KGA, CELF, and the SAs do not need multiple simulations. We test the performance of these algorithms and the GA and discuss features of their solutions.



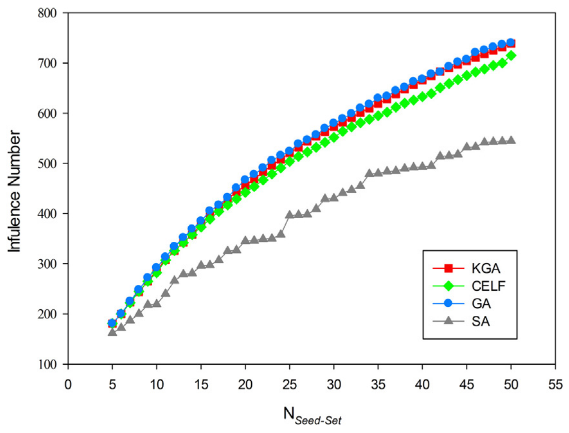
(a) NETSCIENCE network.



(b) EMAIL network.



(c) INFECTIOUS network.



(d) UCSOCIAL network.

Fig. 3. Comparing influence number with four testing algorithms in (a) NETSCIENCE network, (b) EMAIL network, (c) INFECTIOUS network and (d) UCSOCIAL network. $N_{seed\text{-}set}$ is the size of seed set. Each point is the influence number with an algorithm under a specified size of the seed set. Each point on a curve is the average of 100 runs. The red-square curves represent the results with KGA; the green-diamond curves represent the results with CELF; the blue-circle curves represent the results with GA; and the gray-triangle curves represent the results with SA. Because the results of KGA and GA in (b) and (c) are very close, the points in curves overlap.

Table 3
Parameters of the GA in experiments.

Parameters	Value
Population number: m_{pop}	10
Clone population number: n_{pop}	10
Mutation probability: p_m	0.8
Crossover probability: p_c	0.2

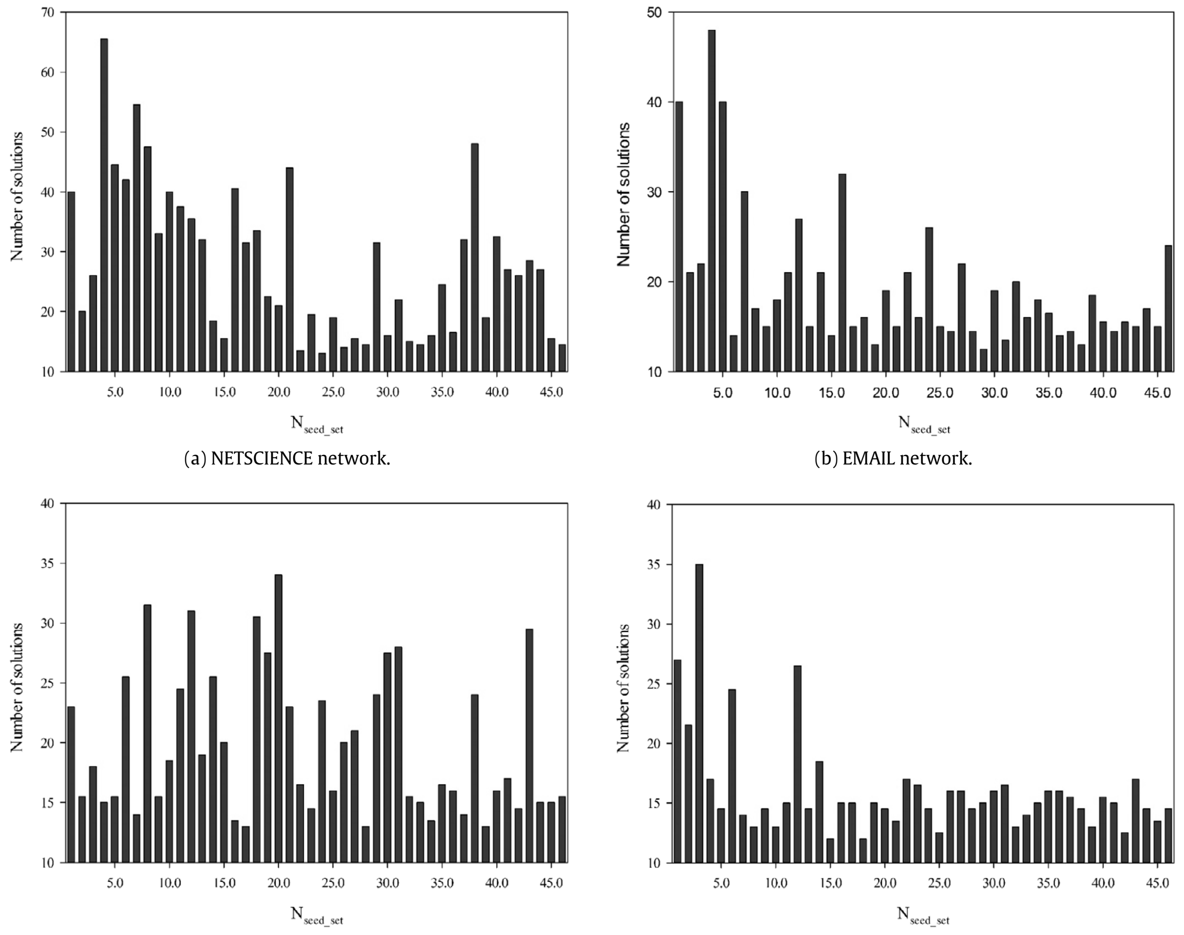
In the second experiment, we test the influence of crossover probability p_c and mutation probability p_m in generated random networks with connection probability 0.15 and size 500. In both experiments, we generate a fixed choice of thresholds, and θ_v follows a uniform distribution on [0,1].

The parameters of the GA are shown in Table 3.

The experimental environment is a PC with Intel Core (TM) i5-2400, 3.1 GHz CPU and 4G RAM, and the sizes of all test networks are less than 2000.

4.2. Experimental results and discussion of actual networks

From Fig. 3 it can be concluded that except for the "NETSCIENCE" network, the GA performed just as well as KGA but better than other algorithms. Overall, comparing the results in detail, we see that when the size of the seed set is small, the



(a) NETSCIENCE network.

(b) EMAIL network.

(c) INFECTIOUS network.

(d) UCSOCIAL network.

Fig. 4. Number of solutions with GA in (a) NETSCIENCE network, (b) EMAIL network, (c) INFECTIOUS network and (d) UCSOCIAL network with population number m_{pop} =100. $N_{seedset}$ is the size of seed sets. Each bar represents the number of solutions under the current seed set.

Table 4
Runtime (s) of different algorithms.

$N_{seed\text{-}set}$	KGA		CELF		GA		SA	
	10	30	10	30	10	30	10	30
NETSCIENCE	397.59	904.86	43.09	146.89	165.89	482.59	150.27	317.36
EMAIL	896.52	2169.54	132.21	409.01	339.32	986.17	254.24	840.44
INFECTIOUS	912.45	2820.30	130.35	331.80	314.28	1034.11	252.08	814.59
UCSOCIAL	2638.48	11568.72	362.43	1217.76	1153.48	1764.05	824.87	1431.06

difference between our results and other algorithms' is not obvious. But with an enlarged seed set, the results of GA are much better than those of other algorithms except for KGA. In the "EMAIL" network when the size of seed set $N_{seed\text{-}set}$ is greater than 10, there is a significant increase in the number of influence nodes. When $N_{seed\text{-}set} > 30$, GA performs just as well as the KGA. In "INFECTIOUS", although the results from GA are quite similar to those of KGA, when $N_{seed\text{-}set} < 30$, GA is always better. In "UCSOCIAL", when $N_{seed\text{-}set} > 16$, GA performs better than the other algorithms. It was not expected that in "NETSCIENCE" GA would fail; the structure of "NETSCIENCE" is sparse, and its network contains a large number of disconnected nodes or small communities. So the heuristic for searching the neighbor set, particularly in the mutation operator, does not work well.

In terms of efficiency, we compare the runtime of the four algorithms when $N_{seed\text{-}set} = 10$ and $N_{seed\text{-}set} = 30$ on the four networks. As the thresholds in the four networks are fixed, we set the rounds of simulation $R = 1$ in KGA, CELF and SA. Table 4 shows the results.

As shown in Table 4, the runtimes of GA are slower than that of CELF and SA, but faster than KGA with $R = 1$ rounds of simulation. However, if the thresholds in networks are not fixed, the rounds of simulation R in KGA, CELF and SA should be

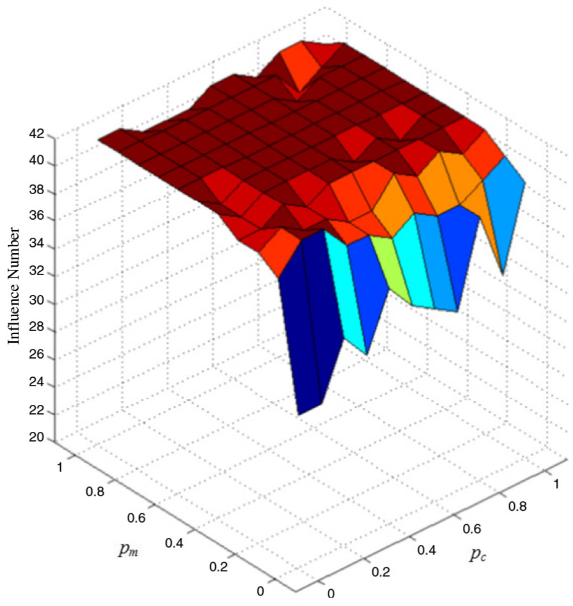


Fig. 5. The average influence node number with different parameters in GA when $N_{seedset} = 10$.

Table 5
The number of influence nodes with different parameters in GA when $N_{seed\_set} = 10$.

	$p_m = 0.1$	$p_m = 0.3$	$p_m = 0.5$	$p_m = 0.7$	$p_m = 0.9$
$p_c = 0.1$	39	41	41	40	41
$p_c = 0.3$	39	41	41	40	41
$p_c = 0.5$	37	41	41	41	41
$p_c = 0.7$	38	41	41	40	41
$p_c = 0.9$	38	41	41	41	41

reset to more than 10,000, which may increase the runtime of those algorithms significantly. Thus GA outperforms those algorithms in efficiency.

For solution diversity, GA has a considerable advantage. As mentioned above, with a fixed choice of thresholds, the influence spread is also fixed. Therefore, KGA and CELF, which have no random mechanism to select the best contributing node, can only obtain one solution. SA has the same problem. But in the GA for different networks, there are always a number of different solution sets. As shown in Fig. 4, GA can always obtain more than two solutions. Another phenomenon seen in Fig. 4 is that when $N_{seed\text{-}set}$ becomes large, the number of solutions becomes smaller. This occurs because when $N_{seed\text{-}set}$ is small, there are more node combinations having the same influence. But when $N_{seed\text{-}set}$ becomes large, the influential nodes in the network are limited, resulting in fewer combinations. Also, as seen in Fig. 4(c), the distribution of solution numbers is different from those in Fig. 4(a), (b), (d). This shows that the network structure may, to some extent, affect the results of the GA, which can produce diverse solutions under a fixed threshold.

4.3. Parameter test

Fig. 5 shows the influence number of a 10 node seed set via the GA with different values of p_c and p_m and Table 5 shows the numerical results for the test.

When $p_c = p_m = 0$, the influence number is a minimum. When $p_c \geq 0.3$, $p_m \geq 0.4$, the influence number tends to a maximum, and remains stable. When $p_c \geq 0.8$, $p_m \geq 0.7$, the influence number has declined slightly, but the result is generally stable. This shows that the parameters p_c and p_m affect the genetic algorithm results in some cases; when these values are small, the GA heuristic fails, and the result of the genetic algorithm in each iteration is similar to that of random search. Increasing the two parameters, the heuristics come into play, and the result is significantly improved. But when the two parameters are close to 1, the two operators may influence each other, and the results of the algorithm will be affected accordingly. In Fig. 5, we also see that the mutation operation is more important than the crossover operation. When p_m is small ($p_m < 0.3$), no matter what the value of p_c, the influence number is always small. On the other hand, we know from previous studies [19] that a high p_c may increase the diversity of the population, which will not help the GA to obtain more seed sets. Therefore, the GA should be used with a moderate p_c and a high p_m.

5. Concluding remarks

We introduced a genetic algorithm, with competition among, and evolution of populations to obtain solutions for the IM problem. The GA can not only obtain better results but also improve the efficiency of the algorithm. From the experiments, we find that the performance of GA is similar to KGA and even better than other methods or algorithms. In addition, in some cases the GA can obtain a number of different solutions, ensuring the diversity of solutions in LT with a fixed threshold. Although the performance of GA is affected by its parameters, its results are much better than those of SA.

In some cases our GA is affected by network structure. Although the GA ensures the diversity of the solution, the process of optimization becomes more complex. In future work, we will focus on adjusting the population structure based on the features of the network and influence spread.

References

[1] N.T. Bailey, The Mathematical Theory of Infectious Diseases, Griffi, London, 1975.
[2] J.D. Murray, Mathematical Biology, Springer Verlag, Berlin, 1993.
[3] D.H. Zanette, Dynamics of rumor propagation on small-world networks, Phys. Rev. E (3) 65 (2002) 1–9.
[4] R. Pastor-Satorras, Vespignani a. Epidemic dynamics and endemic states in complex networks, Phys. Rev. E (3) 63 (2001) 066117.
[5] F.M. Bass, A new product growth model for consumer durables, Management 15 (5) (1969) 215–227.
[6] H. Ma, H. Yang, M.R. Lyu, et al., Mining social networks using heat diffusion processes for marketing candidates selection, in: Proceeding of the 17th ACM conference on Information and knowledge management, 2008, pp. 233–242.
[7] 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, 2001, pp. 57–66.
[8] M. Richardson, P. Domingos, Mining knowledge-sharing sites for viral marketing, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD 02, vol. 02 (3), 2002, p. 61.
[9] D. Kempe, J. Kleinberg, Maximizing the spread of influence through a social network, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, ACM, 2003, pp. 137–146.
[10] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '09, vol. 67 (1), Paris, 2009, p. 199.
[11] J. Leskovec, A. Krause, C. Guestrin, et al., Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'07, ACM, New York, 2007, pp. 420–429.
[12] Q. Jiang, G. Song, G. Cong, et al., Simulated annealing based influence maximization in social networks, in: Twenty-Fifth AAAI Conference on Artificial Intelligence Simulated, 2011(Ic), pp. 127–132.
[13] Y. Wang, C. Gao, G. Song, et al., Community-based greedy algorithm for mining top-K influential nodes in mobile social networks categories and subject descriptors, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'10, ACM, New York, 2010, pp. 1039–1048.
[14] A. Goyal, W. Lu, CELF++: Optimizing the greedy algorithm for influence maximization in social networks, in: Proceedings of the 19th International World Wide Web Conference, 2011, pp. 47–48.
[15] S. Wasserman, K. Faust, D. Iacobucci, et al., Social Network Analysis: Methods and Applications, Cambridge University Press, Cambridge, 1994.
[16] W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '09, vol. 67 (1), Paris, 2009, pp. 199.
[17] W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in: Proceedings - IEEE International Conference on Data Mining, ICDM, 2010, pp. 88–97.
[18] A. Goyal, W. Lu, L.V.S. Lakshmanan, SIMPAT: An efficient algorithm for influence maximization under the linear threshold model, in: Data Mining (ICDM), 2011 IEEE 11th International Conference on, IEEE, 2011, pp. 211–220 Ic.
[19] J. Holland, Adaptation in Natural and Artificial Systems, 1974.
[20] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E, Amer. Phys. Soc. 74 (3) (2006) 36104.
[21] R. Michalski, S. Palus, P. Kazienko, Matching organizational structure and social network extracted from email communication, Bus. Inform. Syst. 87 (2011) 197–206.
[22] Sociopatterns.org[EB/OL].2013 http://www.sociopatterns.org/datasets/infectious-sociopatterns-dynamic-contact-networks/.
[23] T. Opsahl, P. Panzarasa, Clustering in Weighted Networks, Social Networks, 2009.