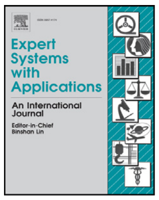
Expert Systems With Applications 55 (2016) 255–267

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Maximizing influence under influence loss constraint in social networks

Yifeng Zeng[a,b,*], Xuefeng Chen[c], Gao Cong[d], Shengchao Qin[e,b], Jing Tang[b], Yanping Xiang[c]

[a] Automation Department, Xiamen University, SIming Road Xiamen, Fujian 361005, China
[b] Teesside University, UK
[c] University of Electronic Science and Technology of China, Chengdu, China
[d] Nanyang Technological University, Singapore
[e] Shenzhen University, China

ARTICLE INFO

Keywords:
Influence maximization
Influence loss
Social networks

ABSTRACT

Influence maximization is a fundamental research problem in social networks. Viral marketing, one of its applications, aims to select a small set of users to adopt a product, so that the *word-of-mouth* effect can subsequently trigger a large cascade of further adoption in social networks. The problem of influence maximization is to select a set of K nodes from a social network so that the spread of influence is maximized over the network. Previous research on mining top-K influential nodes assumes that all of the selected K nodes can propagate the influence as expected. However, some of the selected nodes may not function well in practice, which leads to *influence loss* of top-K nodes. In this paper, we study an alternative influence maximization problem which is naturally motivated by the reliability constraint of nodes in social networks. We aim to find top-K influential nodes given a threshold of influence loss due to the failure of a subset of $R(<K)$ nodes. To solve the new type of influence maximization problem, we propose an approach based on *constrained simulated annealing* and further improve its performance through efficiently estimating the influence loss. We provide experimental results over multiple real-world social networks in support. This research will further support practical applications of social networks in various domains particularly where reliability would be a main concern in a system deployment.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

Social networks provide an intuitive representation about individual connections and display interesting behavioral patterns across various populations of users (Wasserman & Faust, 1994). Social network analysis is attracting more and more attention from different research areas and becomes an important tool for developing intelligent systems in recommendation, crowdsourcing service and so on Domingos and Richardson (2001), Zafarani, Abbasi, and Liu (2014), Sun, Lin, and Xu (2015), Zeng et al. (2015).

The merit of a social network lies in the power of users' interaction that propagates influence of individuals toward the entire network. Such effects have been seen in many real-world applications. For example, a Tweet in *Twitter* is probably followed by hundreds even thousands of the registered users. By exploiting influence spread, a marketing campaign may target a small set of influential individuals and expect that the selected users would generate the largest influence coverage in the market. This is a general problem of influence maximization in social networks where the task is to find top-K influential nodes through influence diffusion models (Kempe, Kleinberg, & Tardos, 2003).

In an ideal circumstance, top-K nodes will spread the influence once they are selected and subsequently activated in a social network. The maximum influence can be achieved only if all of the selected nodes have successfully propagated the influence. However, the influence will be compromised when some of the nodes may not function as they are expected. For example, to market a new product, a company selects a set of retailers that are active and show interest in a similar product market. Due to the changing of financial situations, some of the retailers may not persist the marketing focus on the recommended product. Consequently the new product will not be exposed as much as it should be in the market.

* Corresponding author at: Automation Department, Xiamen University, SIming Road Xiamen, Fujian 361005, China. Tel.: +4401642284311.

E-mail addresses: Y.Zeng@tees.ac.uk (Y. Zeng), cxflovechina@gmail.com (X. Chen), gaocong@ntu.edu.sg (G. Cong), s.qin@tees.ac.uk (S. Qin), X9019186@tees.ac.uk (J. Tang), xiangyanping@gmail.com (Y. Xiang).

http://dx.doi.org/10.1016/j.eswa.2016.01.008
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

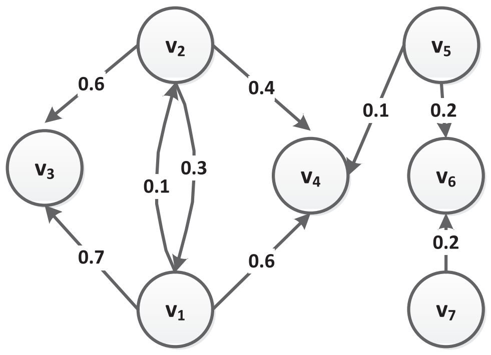


Fig. 1. A social network with influence values on the directed edges. Top-3 influential nodes are $\{v_2, v_5, v_7\}$ in the conventional influence maximization while $\{v_2, v_5, v_1\}$ are solutions to the influence maximization with a tolerable influence loss (1.5).

The *influence loss* occurs to the selected retailers from the perspective of the marketing company. Intuitively, the company may prefer to choose a set of retailers such that they are able to reach a certain level of market coverage (probably not the maximum one), and the market loss is tolerable due to possible malfunctions of the selected retailers in the campaign. As another example application of influence maximization, considering a water network, sensors deployed on the selected locations (pipe junctions) to monitor contaminant spread in the network (Ostfeld, Uber, & Salomons, 2006) as quickly as possible. The detection loss due to the possible malfunctions of sensors would lead to a disastrous effect, and thus it is vital that the detection loss shall be considered when locations are selected. In this paper, we study how the consideration of influence loss would impact the selection of top-*K* influential nodes in social networks.

Things become complex since the set of nodes that are proned to failure are unknown in the selection process. In addition, it is normally hard, if not impossible, to predict individual failure probabilities of nodes in a large scale of social networks, which may depend on many uncertain factors and vary from time to time. On the other hand, it is easier to estimate the number of failure nodes according to previous observations on the malfunctioning networks. For example, in a water network (Ostfeld et al., 2006), it is rather difficult to predict the failure probabilities of individual sensors, which may be affected by their environment, deployment duration, etc. However, it is easier to estimate the number of failure sensors in the future since the sensor quality is a main factor deciding its functions. Similarly, it is difficult to get failure probabilities of retailers in the marking campaign (Hajian & White, 2012) while it is more feasible to predict the number of retailers that may not perform well as expected. Hence we focus on the investigation of influence loss given the number of failure nodes in social networks.

Failure of some nodes may lead to an invisible influence loss while the loss could be significantly large due to failure of others. To act in a pessimistic way, we consider the worst case that the largest influence loss occurs to the selected nodes, and we may tolerate the loss only if it is not beyond a threshold. By computing the influence of nodes through a traditional influence diffusion model, e.g. independent cascade model as well as its improvement (Jacob, Barak, & Eitan, 2001; Liu, Cong, Zeng, Xu, & Meng, 2014a), we elaborate one example of influence loss in Fig. 1.

Example 1. Given $K = 3$, we can compute the influence for the set of nodes, $\{v_2, v_5, v_7\}$, as follows. We first compute the influence for every node in the social network and then sum the individual influence. As v_1 is only influenced by v_2, the influence is 0.3 from the entire set. For the node v_3, two paths, $v_2->v_1->v_3$ and $v_2->v_3$, may spread the influence from the set $\{v_2, v_5, v_7\}$. v_3 could be influenced by either of them or both. Hence, the influence is counted as: 1-(1−0.6) × (1−0.3 × 0.7)=0.684. Similarly, v_4 receives the influence from three paths: 1-(1−0.4) × (1−0.3 × 0.6) × (1−0.1)=0.5572. v_6 gets the influence: 1-(1−0.2) × (1−0.2)=0.36. Finally, the set have deterministic influence (=1) on their own nodes $\{v_2, v_5, v_7\}$. Hence the influence induced by $\{v_2, v_5, v_7\}$ is calculated as: $0.3 + 0.684 + 0.5572 + 0.36 + 3 = 4.9012$.

For the network of a small size, we can compute the influence for every set of three nodes and identify that the set of nodes, $\{v_2, v_5, v_7\}$, exhibit the maximum influence (influence value=4.9012) while nodes, $\{v_1, v_2, v_5\}$, are the second best influential ones (influence value=4.864). Assume that only one node fails in the selected set. We then compute the influence exhibited by the remaining two nodes in the set. The influence difference between the original set of three nodes and the remaining nodes is the influence loss value due to the node failure. Accordingly, the largest loss is 2.4412 for the first set when node v_2 fails; while, the largest loss is 1.2264 when v_1 fails in the second set. Given a tolerable value of influence loss (threshold=1.5), we may accept the second set of nodes, $\{v_1, v_2, v_5\}$, although they are not the top-3 influential ones in the conventional influence maximization problem.

To provide a reliable top-*K* solution to influence maximization problems in social networks, we study influence loss in this paper. Assume that the number of failure nodes could be predicted in influence propagation, we develop an approach to finding top-*K* nodes that maximize the influence spread given a threshold of the largest influence loss. We solve the task of mining top-*K* nodes by formulating it as one constrained optimization problem. In the context of social networks, solving such a problem is rather hard. The greedy algorithm (Kempe et al., 2003) that was extensively used to solve conventional influence maximization problems becomes problematic since feasible/optimal solutions may be prevented. Incrementally adding nodes with the largest marginal influence may simultaneously introduce potential nodes failure of which will result in an incredible influence loss. In addition, the computation of influence loss may further contribute to the solution complexity.

We propose a *Constrained Simulated Annealing* (CSA) (Wah & Wang, 1999; Wang, 2001) based technique for solving influence maximization problems with the constraint of influence loss. The approach is guaranteed to converge towards the optimum with probability one if such solutions exist. *The CSA algorithm development is not trivial in our context* since we need to design a proper penalty function that correctly encodes the influence maximization problem with influence loss constraint in an effective way. More importantly, we need to investigate sufficient conditions and practical parameter settings that guarantee the algorithmic convergence in the new problem.

To improve the efficiency of the CSA algorithm, we develop a new evaluation function that allows a fast computation of influence loss by reusing calculations of individual influence in a social network. Subsequently, we prove that the new algorithm can generate feasible solutions to the complex influence maximization problem. We conduct extensive experiments in multiple real-life social networks and demonstrate performance of the proposed algorithms.

The remainder of this paper is organized as follows. In Section 2, we review the most relevant work on influence maximization techniques. In Section 3, we formulate an influence maximization problem with influence loss constraint and prove its hardness. In Section 4, we propose the CSA based algorithm and further improve its efficiency by developing a new penalty function in CSA. The algorithmic convergence and complexity are also

Table 1
Notations.

Notations	Descriptions
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$	Social network, its set of vertices $\mathcal{V}$ and edges $\mathcal{E}$
N	The number of nodes in $\mathcal{G}$
M	The number of edges in $\mathcal{G}$
S	The seed set
A	The failure set
K	The number of seed nodes
R	The number of failure nodes
P_{vu}	The propagation probability from v to u
$\sigma(S)$	The expected number of nodes influenced by S
η	The influence loss threshold
$h(v, u)$	The influence spreading path from v to u
$AP(v, u)$	The expected influence on u activated by v
$L_{\Omega}(S, \lambda)$	The penalty function of joint space $\Omega = (S, \lambda)$

analyzed. In Section 5, we conduct a series of experiments to demonstrate the algorithm performance in multiple social networks. Finally, we summarize this research and discuss its limitations followed by some hints on future directions.

2. Related work

We briefly review influence maximization problem as well as relevant techniques and elaborate one state-of-art influence maximization algorithm. Notations used in this paper are summarized in Table 1.

2.1. Influence maximization problem

The problem of influence maximization has received substantial attention in the past decade. Richardson *et al.* (Domingos & Richardson, 2001; Richardson & Domingos, 2002) are pioneers on studying influence maximization problem in social networks. They describe the problem in a probabilistic framework and resolve it using Markov Random Field. Kempe et al. (2003) formulate the problem as a discrete optimization problem. They prove the *NP-hardness* of influence maximization problems, and propose a greedy algorithm to approximately solve it by repeatedly selecting the node incurring the largest marginal influence increase to a seed set.

Most of the subsequent research on mining top-K influential nodes is developed based on the greedy algorithm and improves the algorithm performance by reducing the complexity on calculating influence spread. For example, Leskovec et al. (2007) propose a mechanism called *Cost-Effective Lazy Forward* (CELF) to reduce the number of times required to calculate influence spread. Chen *et al.* propose two fast heuristics algorithms, *DegreeDiscount* (Chen, Wang, & Yang, 2009) and *PMIA* (Chen, Wang, & Wang, 2010), to select nodes at each step of the greedy algorithm. Meanwhile, Wang, Cong, Song, and Xie (2010) solve the problem by exploring the underlying community structure of social networks. Based on this work,Song, Zhou, Wang, and Xie (2014) propose a more efficient divide-and-conquer method. Jiang et al. (2011) resort to a *simulated annealing* (SA) algorithm to find top-K influential nodes. Assuming a small propagation probability, the SA algorithm adopts a simple fitness function by exploiting the property of neighboring nodes of a seed set in social networks. Jung, Heo, and Chen (2012) employ the influence ranking technique to scale up the mining algorithms. Liu et al. (2014b) use the parallel processing capability of GPU to accelerate the process of solving influence maximization problem. Liu et al. (2014a) propose the *Influence Spread Path* (ISP) technique for computing influence spread. Following the similar strategy, Kim, Kim, and Yu (2013) present an independent path algorithm and parallelize the algorithm in order to efficiently solve some large social networks. However, it seems that there is no straightforward way to generalize the mentioned algorithms for solving the new influence maximization problem in this paper.

Recently, the influence maximization problem with impact factors emerges as a new line of research on social networks (Chen, Lu, & Zhang, 2012; Gomez-Rodriguez & Scholkopf, 2012). Goyal, Bonchi, Lakshmanan, and Venkatasubramanian (2013) investigate the problem on searching the smallest set of influential nodes whose expected spread is beyond a threshold value at a time point. Similarly, Nguyen and Zheng (2012) study the budgeted influence maximization in which activating each node incurs an arbitrary cost. Li, Chen, Feng, Tan, and Li (2014) consider the influence maximization with location limitation and propose two greedy algorithms for solving their problem. Feng et al. (2014) investigate the effect of novelty decay on influence propagation on real-life datasets and solve the influence maximization with novelty decay. The aforementioned solutions mainly extend the greedy algorithm to solve the influence maximization problem with constraints. The algorithms select candidate nodes by making a trade-off between their marginal influence increase and resulting cost to the seed set. However, influence loss studied in this paper can only be computed when an entire selection process is completed. This prevents an immediate extension of the greedy algorithm with a guarantee of the solution quality.

2.2. Influence spread path methods

The ISP method (Liu, Cong, Xu, & Zeng, 2012; Liu et al., 2014a) provides a structural representation that can speed up the calculation of influence spread. An influence spreading path, $h(v, u)$, is from one of the seed nodes, $v \in S$, to a non-seed node, $u \in (\mathcal{V} - S)$, and the path does not contain any of the other seed nodes in between. The path probability, $P_{h(v, u)}$, is the multiplication of the propagation probability of all edges in the path, e.g., $P_{h(v,u)} = \prod_{(v',u') \in h(v,u)} P(v', u')$. Subsequently, we may calculate the expected influence, $AP(S, u) = 1 - \prod_{v \in S}(1 - P_{h_{(v,u)}})$, on a non-seed node u activated by the seed set S, by assuming that all the paths ending at u are independent. Thus, the influence of the seed set can then be estimated as the sum of all $AP(S, u)$ over the non-seed set, e.g., $\sigma(S) = \sum_{u \in (\mathcal{V}-S)} AP(S, u)$.

There exist a host of approaches to estimating influence spread for a given seed set. It is not our interest to propose another method. In this paper, we employ the ISP method (Liu et al., 2014a) in the proposed CSA algorithms. It is demonstrated in the experiments that ISP achieves good performance (with a trade-off between influence spread and run time) compared to a wide range of existing algorithms. In this paper, we will integrate the ISP method into the proposed CSA algorithms by employing it to estimate influence of the set of seed nodes. However, we shall note that other existing influence calculation techniques can also be integrated into our proposed CSA algorithms for the same purpose although they cannot be directly used to solve the new influence maximization problem.

3. Problem formulation

Consider a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $|\mathcal{V}| = N$ vertices and $|\mathcal{E}| = M$ edges. For each edge $(v, u) \in \mathcal{E}, P_{vu}$ is the probability of influence being propagated over the edge. A conventional influence maximization problem is to find a seed set $S \subseteq \mathcal{V}$ given $|S| = K$ such that the influence, $\sigma(S)$, is maximized according to a diffusion model. We adopt *independent cascade* (IC) model (Kempe et al., 2003) that is widely used in the literature. Given an initial set of seed nodes, S, the IC model propagates influence in inductive

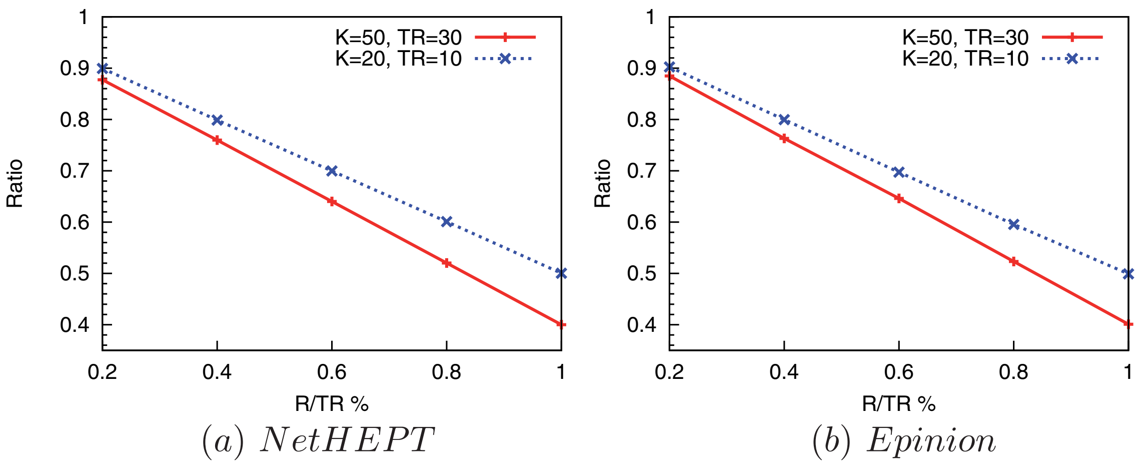


Fig. 2. The ratio of the average influence spread of $(\mathcal{S}-A)$ to the influence spread of top-K nodes decreases when more nodes fail in the seed set (S) instead of the non-seed set $(\mathcal{V}-S)$.

steps. Let Z_t be the set of nodes activated at t, and $Z_0 = S$. Every *active* node v $(\in Z_t)$ has a single chance to activate any of its currently *inactive* node, $u \notin \bigcup_0^t Z_t$, with the probability P_{vu}. The propagation process terminates if and only if we cannot find any active node. The expected number of all active nodes is denoted as $\sigma(S) = \sum_0^\infty |Z_t|$.

Let A be the set of failure nodes of size $R(=|A|)$ in the influence propagation. Influence loss occurs when the set of nodes fail in propagating the influence as they are expected. The selected R nodes may work properly in an initial setting, but fail to function later in an actual propagation. In general, nodes in any part of a network could fail; however, they may lead to more visible influence loss when nodes in a seed set S fail. Let TR be the total number of failure nodes and R be the number of failure seed nodes. We compute $Ratio = \frac{Ave[\sigma(\mathcal{S}-A)]}{\sigma(S)}$, where $Ave[\sigma(\mathcal{S}-A)]$ is the average influence spread when TR nodes, including both R nodes in top-K nodes (S) and $(TR-R)$ nodes in $(\mathcal{V}-S)$, fail in a network once the seed set S has been selected. We conduct experiments with the computation in two publicly available real-world social networks: *NetHEPT* and *Epinion*. Details about the networks will be elaborated in Section 5.

Fig. 2 shows that the influence ratio decreases when the percentage $\frac{R}{TR}\%$ increases (more nodes fail in the seed set instead of the non-seed set). In both cases (K =50 or 20), the influence spread is reduced by half (even more) if all of the failure nodes are selected from the seed set ($\frac{R}{TR}\% = 1$). Hence, focusing on the seed set may prevent unaffordable influence loss due to the node failure.

In this paper, we consider the failure set as a subset of seed nodes, $A \subseteq S$, where in general $R \leq K$. We define influence loss as the reduced influence $[\sigma(S) - \sigma(S-A)]$, where $\sigma(S-A)$ is the expected influence of nodes excluding the failure nodes. Essentially we need to calculate the influence of $(K-R)$ nodes since a set of R nodes are to be removed from the network due to their failure. The failure of some R nodes may not largely impact the expected influence of K nodes while a significant influence may be lost due to the failure of other R nodes. We consider the worst case: the largest loss, *MaxLoss*, occurs when the maximum influence of the remaining $(K-R)$ nodes is the minimum one compared to that of any set of other $(K-R)$ nodes. Formally, given $|A| = R$, $MaxLoss = \sigma(S) - min_{A\subseteq S}\sigma(S-A)$.

For a given social network, our problem is to find a seed set S of size K that maximizes $\sigma(S)$ subject to that the largest influence loss is not above a threshold $\eta(>0)$ due to a failure set A of size R. We may formulate the *Influence Maximization problem with Influence Loss constraint* (IMIL) as a constrained optimization problem below.

$$\begin{aligned} \textbf{Given}: &\quad \mathcal{G}, K, R, \eta \\ \textbf{Objective}: &\quad \max_{S\subseteq\mathcal{V},|S|=K} \sigma(S) \\ \textbf{Constraint}: &\quad \sigma(S) - \min_{A\subseteq S,|A|=R} \sigma(S-A) \leq \eta \end{aligned} \tag{1}$$

Note that a sufficiently large value of η may convert the IMIL problem into a conventional influence maximization problem without constraint (IM). Since the conventional IM problem is *NP-hard*, we may prove that the IMIL problem is *NP-hard*, which is given in Theorem 1.

Theorem 1. *The influence maximization problem with influence loss constraint (IMIL) is NP-hard.*

Proof. Given a conventional influence maximization (IM) problem instance φ: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and K, we can construct an IMIL instance ω by adding an influence loss constraint, $\sigma(S) - min_{A\subseteq S}\sigma(S-A) \leq \eta$ and $\eta \geq N$. As $\sigma(S) \leq N$, every S meets the influence loss constraint. Hence, S are the top-K influential nodes of ω iff S is the selected seed set of φ. As the IM problem has been proved to be *NP-hard* (Kempe et al., 2003), the IMIL problem is *NP-hard* as well. □

We observe that the IMIL problem requires the settings of K, R and η for a given network. In general, K refers to the limited budget in the marketing campaign while R is the amount of campaigning resource that may fail to meet the expectation on propagating influence in practice. η is tolerable loss of the market coverage and mainly depends on personal preference, particularly the risk profile, over the expected market coverage. A reasonable setting of R and η ensures feasible solutions to the IMIL problem, which is assumed in this paper. On the other hand, we will examine the impact of the parameter settings on the proposed algorithms in the following sections.

4. Our methods

We first show that the greedy algorithm for conventional IM problems may not generate feasible solutions to the IMIL problem. Subsequently, we develop a Constrained Simulated Annealing (CSA) based algorithm for solving IMIL. Meanwhile, we propose an approach to improving the efficiency of the CSA algorithm and prove feasibility of the new algorithm.

4.1. Infeasibility of plain greedy algorithms

A natural solution to the IMIL problem attempts a plain greedy algorithm that is the cornerstone of many state-of-art techniques for influence maximization (Kempe et al., 2003). This plain greedy algorithm is one of the most important approximation techniques, with a lower bound ratio of $1-\frac{1}{e}$. Its principle is to repeatedly add the node incurring the largest marginal influence increase to the seed set *S* until the size of the seed set reaches *K*.

As the plain greedy algorithm picks up a node without considering possible influence loss of its own or other nodes, it may lead to a large amount of influence loss when some of the selected nodes fail in the actual influence propagation. This is mainly because the newly selected nodes (into the top-*K* nodes) do not preserve the joint influence of *K*-1 nodes if some of the nodes would fail. Consequently, the algorithm may not generate a feasible solution even if such solutions exist in the IMIL problem. For example, using the greedy algorithm to solve the IMIL problem ($K = 3$ and $R = 1$) in Fig. 1, we first pick the node, v_2, and then v_5. After that, the algorithm selects the node v_7 as it has the largest marginal influence increase to the incomplete seed set $\{v_2, v_5\}$, which generates the maximum influence of the top-3 nodes. However, the seed set of $\{v_2, v_5, v_7\}$ is not a feasible solution when influence loss (given the threshold 1.5) is considered in the influence maximization problem. As we know, the optimal selection shall be the set $\{v_2, v_5, v_1\}$ as including v_1 will compensate the influence loss due to the failure of v_2.

Another attempt is to select top-$(K - R)$ nodes, which explicitly considers the effect of *R* failure nodes, through existing algorithms for conventional influence maximization. However, since the set of failure nodes is unknown, it is impossible to select $(K - R)$ nodes that guarantee to be successful on propagating the influence. Hence it is not proper to simply convert our problem into the issue on finding top-$(K - R)$ nodes.

To seek feasible solutions to the IMIL problem, we extend the greedy algorithm by employing one backtracking strategy in the search. The extended algorithm, called *GreedyB*, first generates top-*K* influential nodes. Subsequently, it updates the top-*K* nodes if the seed set does not meet the constraint. For each update, only one candidate node is replaced with the one having the largest marginal influence increase among the rest $(\mathcal{V} - S)$ nodes. The procedure is terminated once a new solution satisfies the constraint.

Built upon the backtracking strategy, the *GreedyB* algorithm generates a feasible solution to the IMIL problem; however, the solution optimality is not guaranteed in a theoretical way since the replacement process does not optimize the solution under the constraint. Similar to some extension of greedy algorithms (Goyal et al., 2013; Nguyen & Zheng, 2012), one possible way would select candidate nodes by making a trade-off between their influence increase and influence loss to the seed set. However, the information loss is subject to the changing of the seed set and cannot be computed without completing the entire selection process. This motivates us to develop a new technique for solving the IMIL problem.

4.2. CSA based approaches

Essentially we need an algorithm that can strategically explore an entire solution space of a seed set and accept candidate solutions in a systematic way. We proceed to develop a Constrained Simulated Annealing (CSA) based algorithm for solving IMIL. The CSA technique provides a uniform way to handle both objective functions (influence maximization) and constraints (influence loss) in the search space.

CSA extends conventional SA (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953) to solve constrained nonlinear programming problems (Wah & Wang, 1999). Particularly we resort to the CSA method for discrete constrained optimization problems. CSA searches solutions to a penalty function that is a summation of its objective and constraint functions weighted by a penalty factor. Different from SA, CSA looks for saddle points in its search space instead of locating a local extremum.

```
CSA for IMIL
Input: $\mathcal{G}$, $K$, $R$, $\eta$, initial temperature $T_0$,
termination temperature $T_f$ and trial number $q$
Output: Seed set $S$, where $|S|=K$
1. Initialize $t \leftarrow 0$, $T_t \leftarrow T_0$, $c \leftarrow 0$
2. Initialize a solution $D \leftarrow (S,\lambda)$ including $\lambda=0$
   and a seed set $S \subseteq \mathcal{V}$ with $(|S|=K)$
3. While $T_t > T_f$ do
4.      Compute $L_D(S,\lambda)$
5.      Update a neighbour set $N_D$ for $D$
6.      Generate a trail point $D' \in N_D$ randomly
7.      Compute the change of $L$: $\Delta_L \leftarrow L_{D'} - L_D$
8.      Set $c \leftarrow c+1$
9.      If $\lambda' = \lambda$ then
10.       If $\Delta_L<0$ then
11.         $D \leftarrow D'$
12.       else
13.         Generate a random number $\theta \in (0,1)$
14.         If $exp(-\frac{\Delta_L}{T_t}) > \theta$ then
15.           $D \leftarrow D'$
16.     else
17.       If $S' = S$ then
18.         If $\Delta_L>0$ then
19.           $D \leftarrow D'$
20.         else
21.           Generate a random number $\theta \in (0,1)$
22.           If $exp(-\frac{\Delta_L}{T_t}) > \theta$ then
23.             $D \leftarrow D'$
24.     If $c>q$ then
25.       Compute the trial temperature $\rho$
26.       $t \leftarrow t+1$, $T_t \leftarrow \frac{q\times\rho}{ln(t+1)}$, $c \leftarrow 0$
27.Return $S$
```

Fig. 3. The CSA algorithm searches the joint space (S, λ) and computes the penalty function. It will be improved by using a new penalty function in Section 4.3.

4.2.1. Algorithm description

In the IMIL context (formulated in (1)), we first transfer the *max* objective function into the *min* one, and then convert the inequality constraint function into the equality one by using a *max* function,

$max[0, \sigma(S) - min_{A\subseteq S}\sigma(S-A) - \eta] = 0$. The new equality constraint is satisfied iff *MaxLoss* $\leq \eta$. Subsequently, the penalty function is designed in Eq (2).

$$L_{\Omega}(S, \lambda) = -\sigma(S) + \lambda \times max[0, \sigma(S) - min\,\sigma(S-A) - \eta] \quad (2)$$

where Ω is a joint space of seed nodes *S* and a penalty factor λ. A state (trial point) *D* is a solution in the space, e.g., $D = (S, \lambda) \in \Omega$.

A selection of λ values varies on the problem domain. In the IMIL context, we use a relatively large value so that the penalty becomes large once the constraint is broken.

We outline the CSA algorithm for solving the IMIL problem in Fig. 3. In principle, the CSA searches solutions in the joint space Ω for a number of trials per temperature and accepts a solution probabilistically in a cooling schedule. It starts with an initial solution *D* and temperature T_0 (lines 1–2). By computing $\sigma(S)$ and $min_{A\subseteq S}\sigma(S-A)$, the algorithm gets the penalty value L_D for the solution (line 4).

Lines 5–6 compute the penalty function for a neighbor of the current solution *D* in the search space. A neighbour set, N_D, is generated by listing neighboring nodes of *S* and adjacent values of λ.

We randomly sample a new solution D' from N_D. Subsequently, we calculate difference of the penalty functions, Δ_L, between the solutions D and D' (line 7).

Lines 9–23 proceed to check whether the new solution shall be accepted. The algorithm considers the acceptance differently in the subspaces λ and S. If λ has not been changed, we tend to accept the new solution with a lower penalty value that in turn returns a larger $\sigma(S)$. We only accept the solution having a larger penalty value with the probability $exp(-\frac{\Delta_L}{T_t})$ that decreases over time (lines 10–15). On the contrary, we accept the new solution that has a larger penalty value if the seed nodes, S, are not updated (lines 17–23). By doing this, we can increase the penalties of violated constraints, which will enforce S to satisfy the constraints. After we keep searching solutions for q times at a level of temperature T_t, we compute a new temperature (explained later) and start a new trial (lines 25–26). The CSA is terminated when the termination temperature T_f is reached.

4.2.2. Parameter setting

As mentioned in Wah and Wang (1999), the variable S is normally tried 10 times more often than every penalty factor λ. We determine the discrete space for λ, $\Lambda(\lambda)$=[0, $\frac{max\,\sigma(S)}{10\%K*(N-K)}$, $2\frac{max\,\sigma(S)}{10\%K*(N-K)}$, $\cdots$, $max\sigma(S)$], since the maximum number of S's neighbors is $K*(N-K)$. The parameter ρ is one factor of the step temperature (T_t) in the cooling schedule (line 25 in Fig. 3) of CSA. The temperature is reset once a set of q trials have been completed. Also, ρ is relevant to the *asymptotic convergence* (convergence to a global extremum with probability one given $t \to \infty$) as ρ dominates the temperature value of every iteration in the CSA. We let $\rho = 2max\,\sigma(S)(1 + max\,\sigma(A))$ such that the CSA convergence will be guaranteed (explained below). The remaining problem is on how to compute the values of both $max\sigma(S)$ and $max\sigma(A)$ in an efficient way.

We estimate influence values using the generalized ISP method. As the ISP can compute the influence spread of every seed node, the influence sum of all seed nodes that have the largest influence provides an upper bound to $max\sigma(S)$, e.g., $\sum_{u\in S} AP(\mathcal{V}, u) \geq max\,\sigma(S)$, which is exploiting the property of submodular functions. Similarly we can estimate $max\sigma(A)$. Note that the estimation is not an exact influence spread, but is efficient and sufficient for the parameter setting.

4.2.3. Asymptotic convergence

The CSA approach solves the IMIL problem (via the penalty function $L_\Omega(S, \lambda)$) by adding a penalty factor λ to every state that is a set of seed nodes S. Let $Z_{D,D'}$ be the probability of generating state D' from D's neighbor set N_D. Since the neighbor set is a union of both S and λ, we have $Z_{D,D'} = \frac{1}{K(N-K)+|\Lambda(\lambda)|-1}$, where $|\Lambda(\lambda)|$ is the size of the discrete space of λ, $\Lambda(\lambda)$.

As CSA accepts a new solution differently in the subspaces of S and λ (lines 9–23 in Fig. 3), the probability of accepting D' is

$$F_{D,D'} = \begin{cases} \dfrac{exp(-max\,[0, L_{D'} - L_D])}{T_t}, & D' = (S', \lambda); \\ \dfrac{exp(-max\,[0, L_D - L_{D'}])}{T_t}, & D' = (S, \lambda'). \end{cases}$$

Thus, the transition probability from state D to state D' is

$$I_{D,D'} = \begin{cases} Z_{D,D'}F_{D,D'}, & D' \in N_D; \\ 1 - \sum\limits_{W\in N_D} Z_{D,W}F_{D,W}, & D' = D; \\ 0, & otherwise. \end{cases}$$

As the transition probability differs in the subspaces and the transition occurs over time, the CSA procedure can be modeled using an inhomogeneous Markov chain. Let q be the maximum of the minimum number of transitions required to reach optimum from any state. The q value can always be found if the neighbor set is properly constructed.

In the cooling schedule (line 25 in Fig. 3), we have $T_t = \frac{q\times\rho}{ln(t+1)}$. With the specification, $\rho = 2\ max\,\sigma(S)(1 + max\,\sigma(A))$, we get the following proposition.

Proposition 1. *The temperature setting,* $T_t = \frac{q\times\rho}{ln(t+1)}$, *satisfies the properties:* (**a**) $T_t > T_{t+1}$, (**b**) $\lim_{t\to\infty} T_t = 0$, *and* (**c**) $T_t \geq 2\frac{q}{ln(t+1)} \times max|L_{D'} - L_D|$.

Proof. Properties **a** and **b** are obvious as both q and ρ are constant and T_t is the monotonically decreasing function with t. We compute $max|L_{D'} - L_D|$ for two cases where either S or λ changes in a joint space.

$$\begin{aligned}
&\mathbf{Case\,1}: \mathbf{D}' = (\mathbf{S}', \lambda) \\
&max|L_{D'} - L_D| = |-\sigma(S') + \lambda\times max[0, (\sigma(S') - min\,\sigma(S' - A'))] \\
&-[-\sigma(S) + \lambda \times max[0, (\sigma(S) - min\,\sigma(S - A))]| \\
&for \star\star \sigma(S) - \sigma(S - A) \leq \sigma(A) \star\star \\
&\leq |\sigma(S) + \lambda max\,\sigma(A')|\ for \star\star \lambda \leq max\,\sigma(S) \star\star \\
&\leq max\,\sigma(S) + max\,\sigma(S) \times max\,\sigma(A') \\
&\mathbf{Case\,2}: \mathbf{D}' = (\mathbf{S}, \lambda') \\
&max|L_{D'} - L_D| = |\lambda' \times max[0, (\sigma(S) - min\,\sigma(S - A) - \eta)] \\
&-\lambda \times max[0, (\sigma(S) - min\,\sigma(S - A) - \eta)]| \\
&\leq \lambda' \times max\,\sigma(A) \\
&\leq max\,\sigma(S) \times max\,\sigma(A)
\end{aligned}$$

Since $T_t = \frac{q\times\rho}{ln(t+1)} = \frac{2q\times max\,\sigma(S)(1+max\,\sigma(A))}{ln(t+1)}$, property **c** is fulfilled. □

Consequently, given definitions of the transition probability $I_{D,D'}$ and properties of the decreasing temperature T_t, we can get the asymptotic convergence of the CSA for the IMIL problem by following the proof in Wah and Wang (1999). Formally, we have the theorem below.

Theorem 2. *The CSA based algorithm for the influence maximization problem with the influence loss constraint converges to a constraint global optimum with probability one as* $t \to \infty$.

4.3. Improved CSA: CSA-Q

The CSA algorithm needs to evaluate the penalty function of Eq. 2 in each iteration. Since the penalty function involves the time-consuming computation of influence loss, $[\sigma(S) - min\,\sigma(S - A)]$, the CSA complexity is further exacerbated by the complex function evaluation. This motives us to improve the CSA's efficiency using more tractable computation of influence loss.

The property of the submodular function, $\sigma(S)$, allows us to bound the influence loss as shown in Proposition 2.

Proposition 2. *The information loss,* $[\sigma(S) - min\,\sigma(S - A)]$, *is bounded by* $max\sum_{i=1,v_i\in S}^{R} \sigma(v_i)$, *where R is the number of failure nodes* ($|A|$).

Proof. We aim to prove the inequation: $\sigma(S) - min\,\sigma(S - A) \leq max\sum_{i=1,v_i\in S}^{R} \sigma(v_i)$. As $\sigma(S)$ is submodular, we get $min\,\sigma(S - A) \geq min\,[\sigma(S) - \sigma(A)] = \sigma(S) - max\,\sigma(A)$. Thus, $\sigma(S) - min\,\sigma(S - A) \leq \sigma(S) - [\sigma(S) - max\,\sigma(A)] = max\,\sigma(A)$.

Similarly, we get $max_{A\subseteq S}\,\sigma(A) \leq max\sum_{i=1,v_i\in S}^{R} \sigma(v_i)$, where $max\sum_{i=1,v_i\in S}^{R} \sigma(v_i)$ is the influence sum of the first R nodes that are ranked according to their individual influences. □

Accordingly we may approximate the largest influence loss via computing its upper-bound value $max\sum_{i=1,v_i\in S}^{R} \sigma(v_i)$. By doing this, we make a more protective decision on selecting top-K nodes with the consideration of influence loss. Hence we can get the new penalty function, $L'_{\Omega}(S,\lambda)$, in Eq. 3.

$$L'_{\Omega}(S,\lambda) = -\sigma(S) + \lambda \times max[0, max \sum_{i=1,v_i\in S}^{R} \sigma(v_i) - \eta] \tag{3}$$

We denote the improved CSA algorithm by CSA-Q that adopts the approximate penalty function, $L'_{\Omega}(S,\lambda)$, and expects to solve the IMIL problem much more quickly than the original CSA in Fig. 3. Intuitively the evaluation of $L'_{\Omega}(S,\lambda)$ will become much more efficient because it computes influence spread of individual failure nodes, $v \in A$, instead of every set of the remaining nodes $(S-A)$. In addition, we will achieve significant computational savings as influence spread of individual nodes can be computed at one time and be reused for all iterations.

More importantly, the CSA-Q algorithm can generate feasible solutions to the IMIL problem, which is benefited from using the upper-bound value of influence loss in the new penalty function. Proposition 3 formally states feasibility of the CSA-Q algorithm.

Proposition 3. *Top-K nodes generated by the CSA-Q algorithm guarantee to be feasible for the IMIL problem.*

Proof. For top-K nodes S, the CSA-Q solutions satisfy the constraint, $max \sum_{i=1,v_i\in S}^{R}\sigma(v_i) \le \eta$, since it takes the new penalty function $L'_{\Omega}(S,\lambda)$ in Eq. (3). Using Proposition 2, we get $\sigma(S) - min\,\sigma(S-A) \le \eta$. Hence the solutions are also feasible to the IMIL problem as defined in Section 3. □

Note that the CSA-Q algorithm still shares the convergence property with the CSA, which is stated in Theorem 2, while it may converge to a sub-optimal solution in term of influence spread. We will show later that influence spread achieved by the CSA-Q algorithm is very close to that obtained through the CSA algorithm in the experimental study.

Both the CSA and CSA-Q algorithms provide the principled way to searching top-K nodes in the IMIL problem. In every iteration, they need to calculate the penalty functions (Eqs. (2) and(3)) that involve the computation of influence spread e.g. $\sigma(S)$, $\sigma(v)$ and so on. As mentioned previously, any of the existing influence maximization techniques can be employed to compute the influence spread in the CSA and CSA-Q algorithms. We select the ISP method and briefly describe its application in our algorithms.

The ISP method is used to compute influence spread of a seed set that is an influence sum of all activated nodes in a non-seed set (Liu et al., 2014a). Similarly we may use the method to compute the expected influence on the non-seed set activated by an *individual* seed node $v \in S$, e.g., $\sigma(v) = \sum_{u\in(\mathcal{V}-S)} AP(v,u)$. By doing this, we can rank the seed nodes in term of their expected influence over the non-seed set. The ranking operation could be done on the fly and provides an efficient way to estimate the influence spread of a given seed set. For example, to estimate $max\sigma(A)$, we may pick a set of candidate nodes that have the largest expected influence, $AP(v,\mathcal{V}-A)$, in the list and sum all of the influence.

4.4. Time and space complexities

For every iteration of the CSA algorithm in Fig. 3, we compute both $\sigma(S)$ and $\sigma(S-A)$ using the ISP method. As ISP requires to compute influence of all the influence spreading paths, denoted by *PATH(S)* and *PATH(S–A)*, from the seed set and the non-failure set respectively, the run time takes $\mathcal{O}(|PATH(S)|) + \mathcal{O}(|PATH(S-A)|)$ if the depth-first search is used to find all the paths. The total run time is $qT[\mathcal{O}(|PATH(S)|) + \mathcal{O}(|PATH(S-A)|)]$, where T is the number of iterations (in the *while* loop) to terminate CSA.

CSA-Q avoids to compute influence spreading paths, *PATH(S–A)*, for all iterations. In addition, it reuses the influence spread of individual nodes, *PATH(v)*, that can be computed once in the algorithm. Hence the total run time of CSA-Q is $qT\mathcal{O}(|PATH(S)|) + N\mathcal{O}(|PATH(v)|)$. This achieves a large amount of computational savings on solving the IMIL problem.

Meanwhile, the space complexity of both algorithms is $\mathcal{O}(N + M + |PATH(S)|)$ as we need to store a social network and all the paths in the computation.

Table 2
Statistics of four social networks.

Networks	NetHEPT	Wiki	Epinions	Amazon
Node number	15, 233	7, 115	75.9K	262K
Edge number	58, 891	103K	508.8K	1.23M
Clustering coefficient	0.2089	0.2283	0.0617	0.3123
Average degree	14.54	6.7	11.54	4.77
90% effective diameter	3.8	5	4.7	6.5

5. Experiments

We used four publicly available real-world social networks in the experiment and summarize their statistics in Table 2.

We implemented the following algorithms and compared their performance in terms of influence spread and run time.

• *CSA* and *CSA-Q*: We implemented the *CSA* algorithm as presented in Fig. 3, and the *CSA-Q* algorithm using the new penalty function in Eq. 3. The algorithms generate an initial seed set in a random way and develop a neighbor set of seed nodes without excluding previously selected nodes (as the seed set may also be changing in the iterations). To compute influence spread, we implemented the ISP method, and as suggested in Liu et al. (2014a) we remove the paths that are leading to insignificant influence propagation (that is smaller than a threshold 0.0001 - a tradeoff between influence spread and run time).

- *GreedyB*: We implemented the *GreedyB* algorithm as discussed in Section 4.1. Since the *GreedyB* algorithm can produce feasible solutions to the IMIL problem, it acts as a baseline comparison.
- *Random*: We implemented the *Random* algorithm that is often used in the comparison about influence maximization techniques (Chen et al., 2009). To solve the IMIL problem, the *Random* algorithm picks K nodes randomly from a social network and accepts a solution once it meets an influence loss constraint.

All algorithms are implemented in C++ language, and compiled by gcc 4.7.2 on a Linux PC with a 4-core Intel i7-3770 3.4GHz CPU and 8 GB memory. To compute the number of nodes influenced by the selected seed set of an evaluated algorithm, we apply 20,000 Monte Carlo simulations with the seed set selected by each evaluated method, and the average number of influenced nodes is used as influence spread of the seed set. This evaluation method follows the previous work on influence maximization (Chen et al., 2009).

5.1. Parameter settings

In social networks, the activating probability, P_{vu}, is set by the *weighted cascade policy*, e.g. $P_{vu} = \frac{1}{N_{in(u)}}$ where $N_{in(u)}$ is indegree of u, which is widely used in the existing conventional influence maximization techniques (Chen et al., 2009; Kempe et al., 2003). We pick values of η in the range $[min\sigma(A), max\sigma(A)]$, where $min\sigma(A)$ is the minimum influence spread of a set of R nodes (|A|=R), and both $min\sigma(A)$ and $max\sigma(A)$ can be estimated through the ISP method as described in Section 4.3. In the CSA based algorithms, following the spirit of T_t calculation, the initial temperature is set as $T_0 = 2q \times max\,\sigma(S)(max\,\sigma(A)+1)$. Both $\sigma(S)$ and $\sigma(A)$ values are estimated through the ISP method.

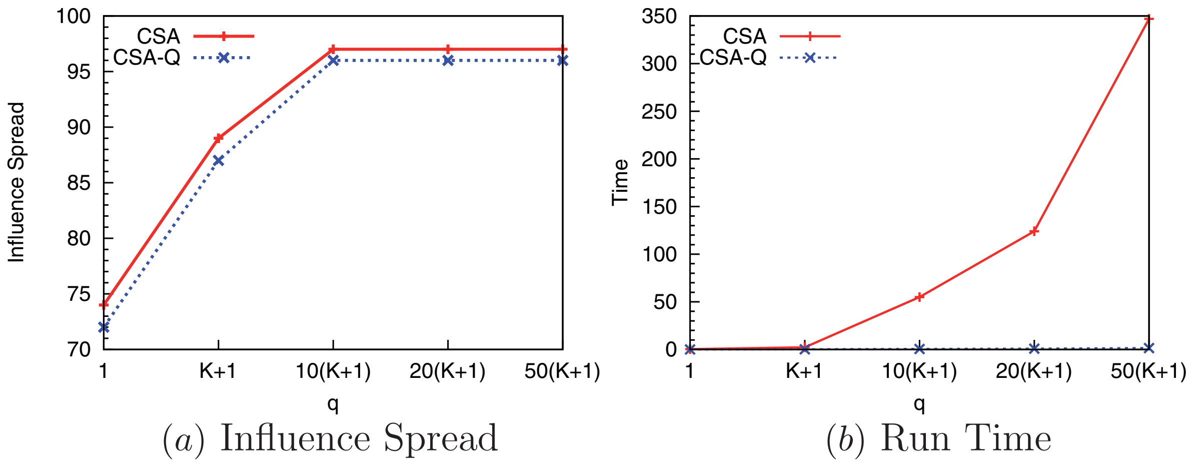


Fig. 4. Selection of q values is a trade-off between influence spread and run time (*mins*). Note the run time of *CSA-Q* is extremely low and the representative curve (colored by blue with cross in (*b*)) nearly overlaps with the q-axis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

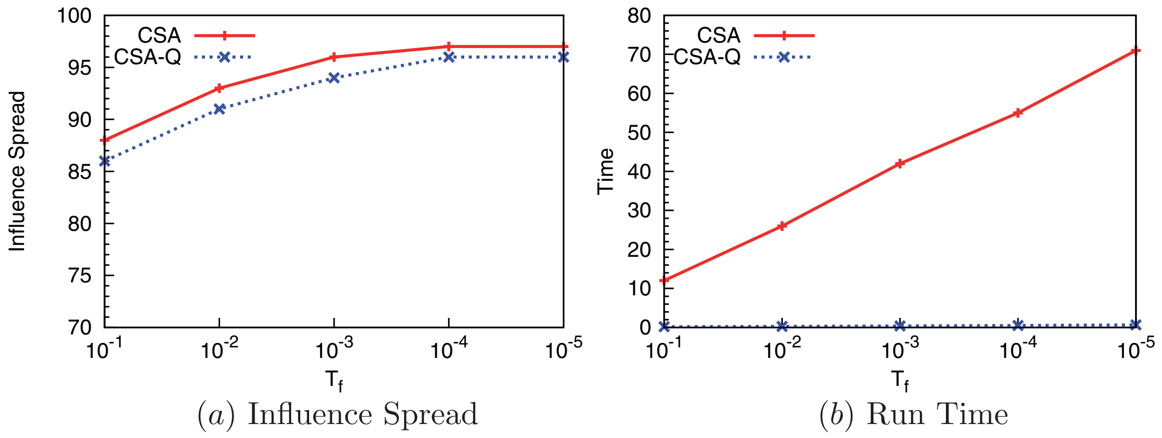


Fig. 5. The termination temperature T_f controls the total iterations of the CSA based algorithms. The lower the termination temperature the more the iterations.

As shown in Fig. 3, the number of trials, q, has a large impact on the run time of the CSA algorithms. Intuitively, few iterations may prevent the algorithmic convergence thereby leading to smaller influence spread. A large number of iterations guarantee the global convergence while incurring a large amount of execution time. Meanwhile, the q value is most relevant to how the neighbor set is constructed and a solution set of K nodes together with a λ value are updated (Lines 5–6 in Fig. 3). Ideally, q is equal to $(K+1)$ if a solution is replaced in an optimal way at each iteration. For a general replacement strategy in the CSA algorithms, we investigate the selection of q values in term of the algorithmic efficiency.

Fig. 4 shows the selection of q values in the *NetHEPT* network with the setting: $\eta = 20$, $K = 10$, $R = 2$, and $T_f = 10^{-4}$. With $q = 10(K+1)$, both the *CSA* and *CSA-Q* algorithms converge to relatively large influence spread and consume a reasonable amount of run time. Since the *CSA-Q* algorithm reuses most of influence spread computation, its run time does not significantly grow with the increasing values of q. In addition, as shown in Fig. 4(*a*), more iterations may not contribute to a visible improvement on performance of both the *CSA* and *CSA-Q* algorithms. Hence we will use $q = 10(K+1)$ in the rest of the experiments.

We take a further step to investigate the setting of the termination temperature T_f that controls the outermost loop of the algorithms in Fig. 3. In Fig. 5, we show influence spread and run time of the CSA based algorithms when T_f values vary in *NetHEPT* with the setting: $\eta = 20$, $K = 10$, $R = 2$, and $q = 10(K+1)$. We observe that both the *CSA* and *CSA-Q* algorithms achieve best cost-effective performance in terms of influence spread and run time when T_f takes the value 10^{-4}.

In summary, we empirically study the parameters of q and T_f in both the *CSA* and *CSA-Q* algorithms, and choose proper values by considering their impact on influence spread and run time in *NetHEPT*. Similar performance is also observed in other networks. We will use $T_f = 10^{-4}$ and $q = 10(K+1)$ in the rest of our experiments.

5.2. Experimental results

With the aforementioned parameter settings, we conduct experiments to demonstrate the performance of our methods.

5.2.1. Influence loss

As mentioned previously, failure of nodes may lead to different levels of influence loss in social networks. We investigate how the failure of different sets of nodes will impact influence spread in *NetHEPT* and *Epinion*. We first employ the ISP method to compute top-K nodes, and then obtain influence spread of (K-R) nodes when any set of R nodes fails in the networks. Subsequently, we compute the ratio of the average influence loss to the influence spread of top-K nodes: $Ratio = \frac{\sigma(S) - Ave[\sigma(S-A)]}{\sigma(S)}$, where $Ave[\sigma(S-A)]$ is the influence spread averaged over P sets of $(K-R)$ nodes randomly

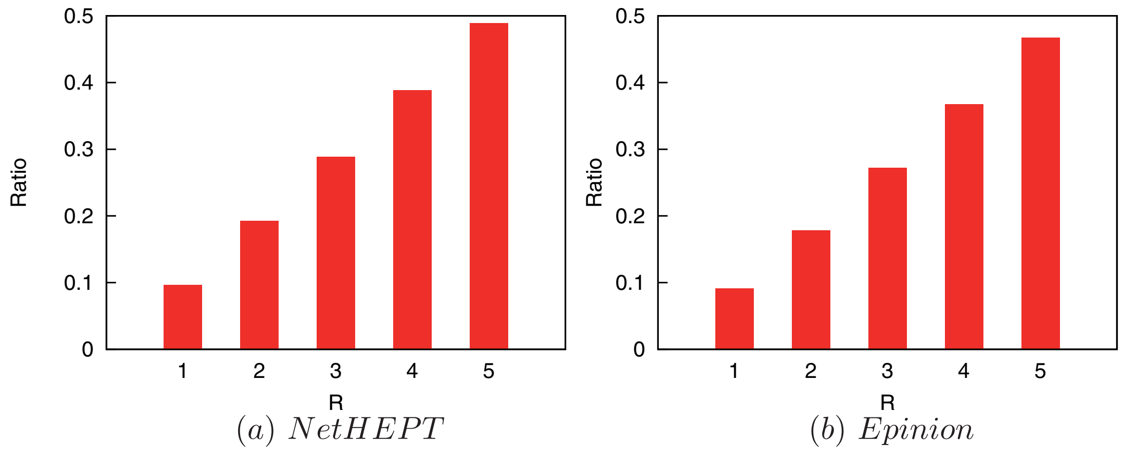


Fig. 6. The ratio of the average influence loss to the maximum influence spread of top-*K* nodes when *R* nodes fail.

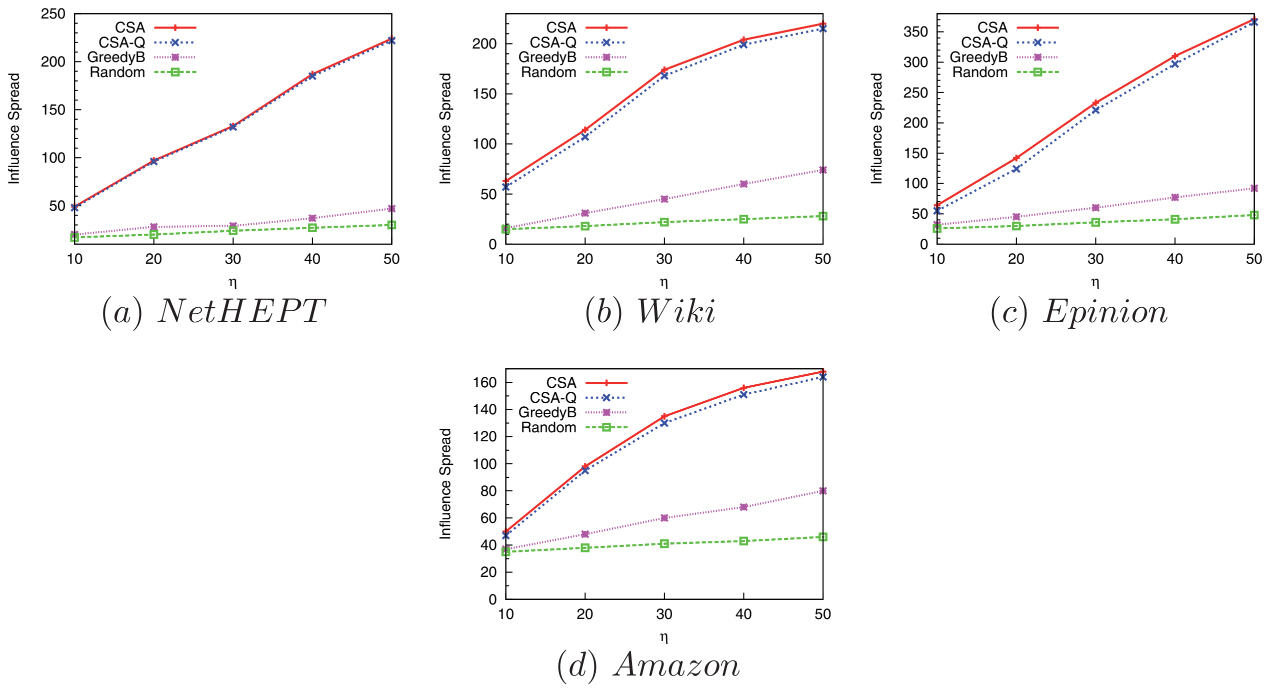


Fig. 7. As η increases, all algorithms achieve more influence spread. In (*a*), the curves of *CSA-Q* and *CSA* overlap in *NetHEPT*.

selected from top-*K* nodes and *P* ranges from tens to thousands depending on the network size and *R* values. Fig. 6 shows the results of ratio in two networks with the setting of $K = 10$ and different *R* values.

In Fig. 6, we observe that the failure of *R* nodes does lead to visible influence loss (around 10% of the maximum influence spread of top-10 nodes) even when only one node fails in the selected seed set. The proportion grows with the increasing number of failure nodes, and rises to around 50% of the total influence spread when half of top-10 nodes fail in both networks. Hence it is important to consider influence loss in influence maximization problems.

5.2.2. Variations of η

We proceed to demonstrate performance of the *CSA* based algorithms given the input parameters including η, *K* and *R*. Fig. 7 shows performance of the algorithms given $K = 10$ and $R = 2$ when η values are varied. A large η value relaxes constraints of influence loss when the failure occurs to the selected nodes. This may retain more influential nodes failure of which may lead to visible influence loss, but still satisfy the loose constraint. As expected, all of the algorithms generate larger influence spread when η increases.

Both the *CSA* and *CSA-Q* methods outperform the other two algorithms in all experiments. The *Random* method performs poorly and does not lead to significant increase on the influence spread even when η increases. Although *GreedyB* may generate feasible solutions, it results in relatively small influence spread compared to the *CSA* and *CSA-Q* algorithms. With a large η value, the *Random* and *GreedyB* algorithms immediately accept any solution that meets the constraint and do not conduct any further search to improve the solution, which generally leads to low influence values.

The *CSA* and *CSA-Q* algorithms continuously improve their solutions with the increasing of η values. The loose constraint, which is ascribed to a large η value, allows both algorithms to search a large space in which more influential nodes could be identified. On the other hand, the performance of the *CSA-Q* algorithm approaches that of the *CSA* algorithm although *CSA-Q* uses an

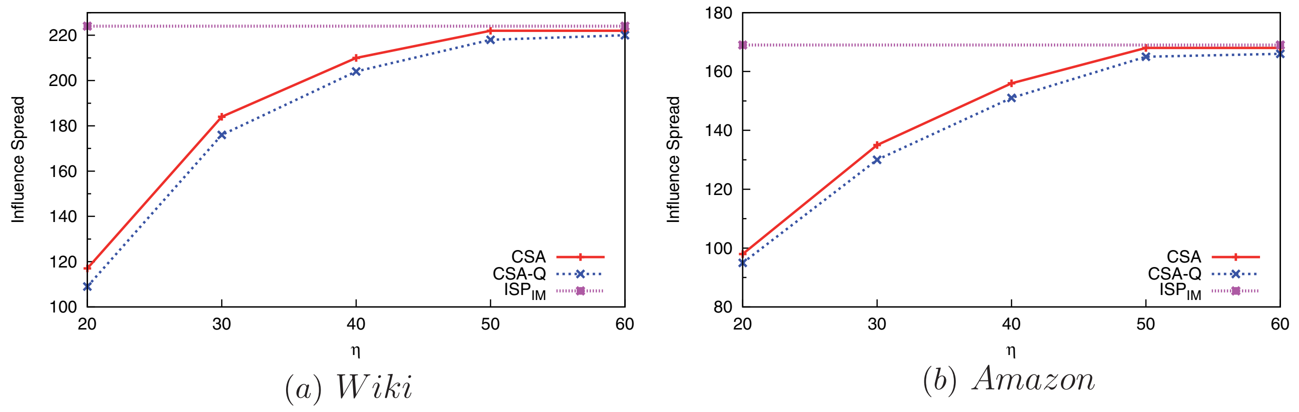


Fig. 8. Performance of the *CSA* and *CSA-Q* methods is similar to that of the ISP_{IM} approach when η is sufficiently large. The top horizontal line denotes the performance of ISP_{IM} when the ISP method is used to solve the conventional IM problem.

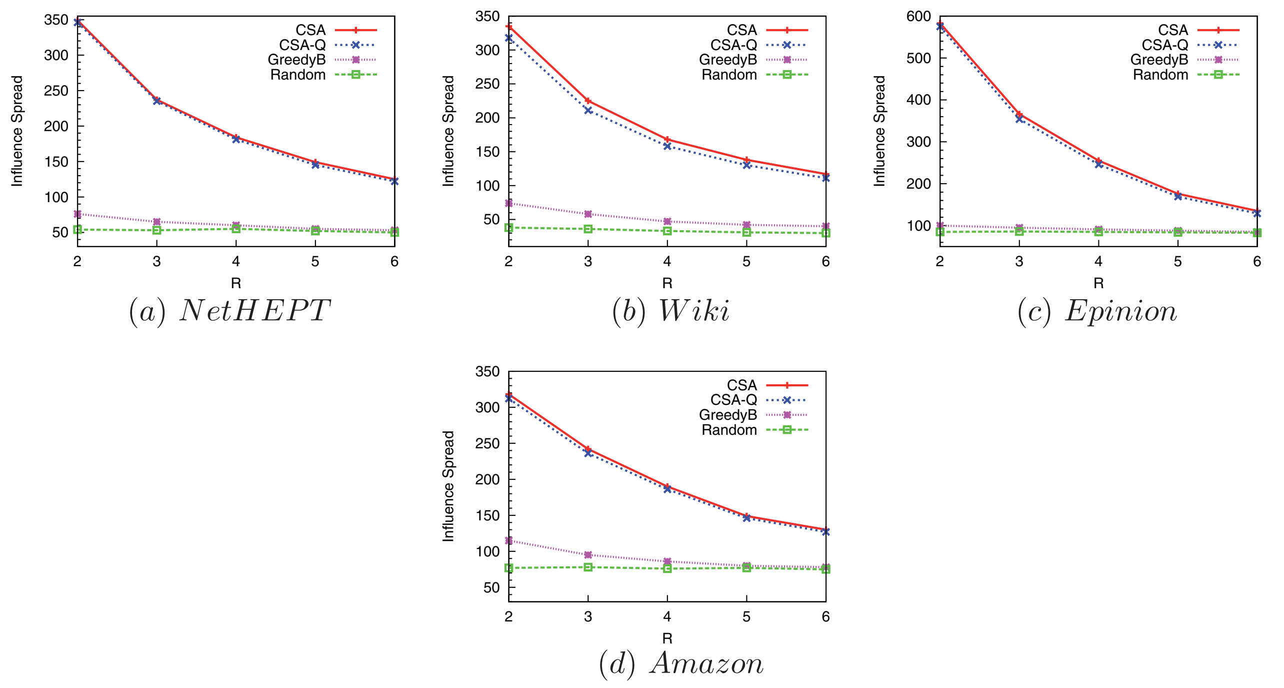


Fig. 9. Influence spread achieved by the algorithms decreases as more nodes (R) fail in the networks ($K = 20$, $\eta = 40$). *CSA-Q* is consistently close to *CSA*.

approximate penalty function. The gap of their performance is extremely small and even becomes invisible in some network like *NetHEPT* in Fig. 7(*a*). It verifies that the new penalty function is good enough to ensure the high quality of solutions achieved by the *CSA-Q* algorithm.

To further verify performance of the *CSA* based framework, we additionally conduct the comparison to the ISP approach, denoted by ISP_{IM}, that is employed to solve conventional influence maximization problems *without influence loss constraints* and to find top-K nodes. In Fig. 8, we show the results, including the reference of ISP_{IM}, in *Wiki* and *Amazon*, with $K = 10$ and $R = 2$. We observe that the influence spread achieved by the *CSA* and *CSA-Q* methods gradually converges to the value obtained by ISP_{IM} when η is larger than 50. Given a sufficiently large η value, the IMIL problem is converted to the conventional IM problem. The solutions returned by the *CSA* based framework are identical to top-K nodes found by ISP_{IM}. Hence both the *CSA* and *CSA-Q* algorithms perform as well as the ISP technique on solving the conventional IM problem.

5.2.3. *Effect of* R *and* K *values*

Fig. 9 shows performance of all the algorithms when R values vary in the experiments. We fix $K = 20$ and $\eta = 40$ for all networks. The joint impact of more failure nodes normally leads to a significant reduction on the influence spread. To prevent such a large influence loss, the algorithms are likely to choose more correlated nodes that may complement each other once the failure occurs to some of the selected nodes. This causes a low value of influence spread achieved by top-K nodes.

In Fig. 9, the influence spread generated by all the algorithms decreases while the number (R) of failure nodes increases. Meanwhile, both the *CSA* and *CSA-Q* algorithms identify better solutions when they are compared to other two algorithms, *GreedyB* and *Random*. The *CSA-Q* algorithm still achieves relatively large influence spread as *CSA* does in four networks. The gap of their performance is even invisible in *NetHEPT* and *Amazon*, which is indicated by the overlapping curves in Figs. 9(*a*) and (*d*) respectively.

Fig. 10 exhibits the influence spread of the top-K nodes returned by four algorithms for different K values. We fix $R = 2$ and

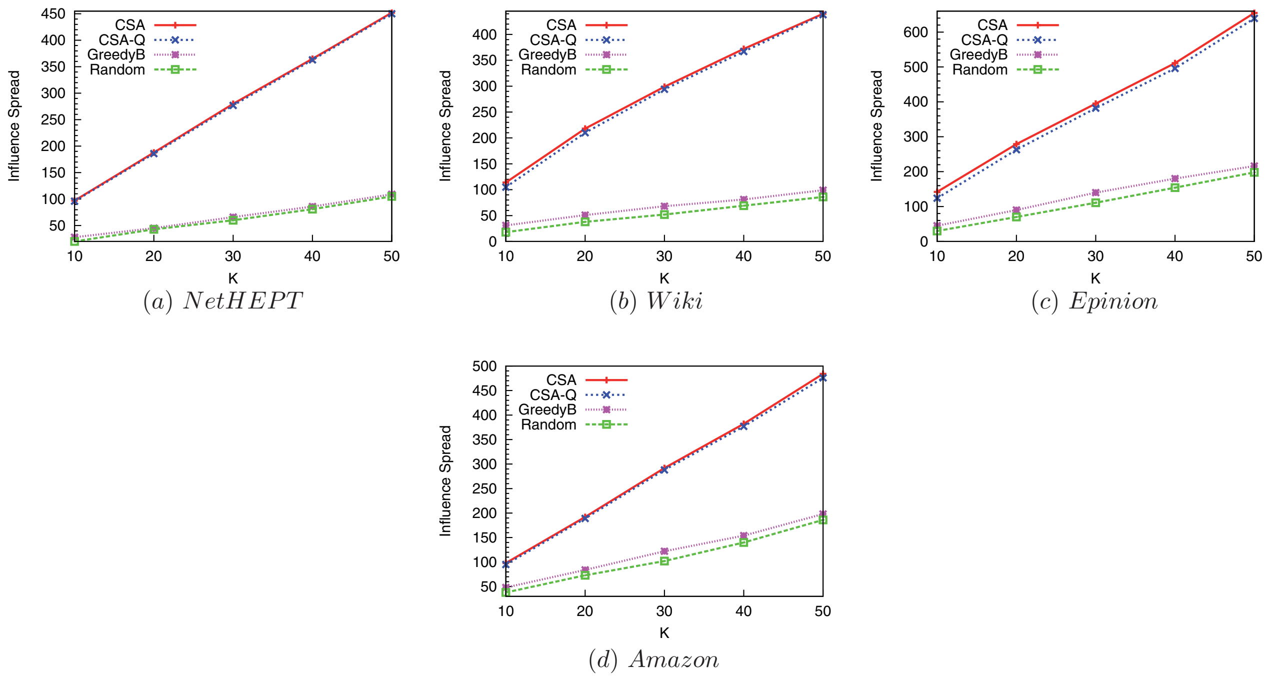


Fig. 10. The algorithms obtain larger influence spread as K increases in four networks ($R = 2$, $\eta = 20$).

$\eta = 20$ for all networks. It is not a surprise that the influence spread increases given a larger K value. Both the *CSA* and *CSA-Q* algorithms consistently outperform the *GreedyB* and *Random* algorithms for different K values over four networks. The *CSA-Q* algorithm achieves similar influence spread as the *CSA* algorithm does in most of the experiments.

5.2.4. Runtime comparison

In Tables 3 and 4, we compare different algorithms based on the run time each takes to identify top-K nodes in four networks. We measure the run time of *Rand.(om)* in milli-seconds(*ms*) or seconds(*s*) while using minutes (*mins*) to measure the time of other three methods. The cell ∗ indicates that the program has run over one day.

Although the *Rand.(om)* method runs fast (measured by *ms* or *s*) in most of cases, it is deemed to generate quite low influence spread (as shown in the experimental results above) due to its randomness. Hence the *Rand.(om)* solutions are not acceptable on solving the IMIL problem. In addition, the *Rand.(om)* method still needs to compute influence loss of every subset of the selected nodes. Consequently, it has to spend much time on solving the cases of large K (or R) in all networks. For example, it costs around 870*mins* to solve the IMIL problem in the network *Wiki* with $K = 50$ and $R = 10$. This is rather time-consuming as the *CSA-Q* algorithm spends only 55.3min on solving the similar case.

The *Gre.(edy)B* algorithm requires a large amount of time to solve the IMIL problem. The additional step of backtracking strategy contributes to its complexity thereby causing more execution times. The algorithm may repeatedly conduct the replacement process to find a feasible solution. Consequently, the *Gre.(edy)B* algorithm cannot solve most of complex cases of large K (or R) in all of four networks.

Both the *CSA* and *CSA-Q* algorithms consume substantially less time compared to the *Gre.(edy)B* algorithm. The savings are mainly ascribed to the systematical search of the CSA based techniques. Particularly the *CSA-Q* algorithm shows significant speed-up over the *CSA* algorithm. Even when either K or R increases, the *CSA-Q* algorithm does not require much more run time. For example, for solving the largest network *Amaz.*(*on*), the increase of run time is less than 10min when K increases from 30 to 50 ($R = 2$, $\eta = 20$).

When η increases given fixed (K, R) values (e.g. (10,2)), both *Rand.(om)* and *Gre.(edy)B* do not cost more time while the *CSA* and *CSA-Q* algorithms require more time to achieve the convergence. It is easier to find feasible solutions when the constraint is relaxed. On the other hand, the loose constraint allows the CSA based algorithms to search better solutions through more iterations.

Table 3
Run time of the algorithms over the networks **NetH.(EPT)** and **Wiki**.

Net.	K	R (η)	Rand.	Gre.B	CSA	CSA-Q
	10	2(20)	9 ms	168	55	0.5
		2(30)	8 ms	168	72	0.6
		2(50)	8 ms	164	125	1.6
		5(50)	22 ms	682	201	0.8
	20	2(20)	60 ms	1358	679	2.1
		2(40)	10 ms	1440	885	3.5
		4(40)	98 ms	∗	1305	2.4
		6(40)	208 ms	∗	∗	1.2
NetH.	**30**	2(20)	259 ms	∗	∗	4.5
		8(80)	27 s	∗	∗	5
	40	2(20)	525 ms	∗	∗	5.5
		10(100)	65 mins	∗	∗	7.0
	50	2(20)	980 ms	∗	∗	8.6
		10(100)	790 mins	∗	∗	11.4
		20(200)	∗	∗	∗	13.2
	10	2(20)	20 ms	512	157	3.3
		2(30)	18 ms	462	242	4.1
		2(50)	16 ms	400	305	6.0
		5(50)	52 ms	1520	271	3.8
	20	2(20)	210 ms	∗	954	12
		2(40)	165 ms	∗	1062	19.5
		4(40)	315 ms	∗	1398	13
		6(40)	615 ms	∗	∗	8.5
Wiki	**30**	2(20)	1.5 s	∗	∗	27
		8(80)	30 s	∗	∗	34.5
	40	2(20)	2 s	∗	∗	36
		10(100)	70 mins	∗	∗	48
	50	2(20)	2.8 s	∗	∗	52
		10(100)	870 mins	∗	∗	55.3
		20(200)	∗	∗	∗	59.6

Table 4
Run time of the algorithms over the networks **Epin.(ions)** and **Amaz.(on)**.

Net.	**K**	**R** (η)	**Rand.**	**Gre.B**	**CSA**	**CSA-Q**
	10	2(20)	33 ms	2850	191	5.5
		2(30)	30 ms	2772	316	5.7
		2(50)	26 ms	2560	504	6.5
		5(50)	81 ms	*	476	6.3
	20	2(20)	500 ms	*	1216	7.8
		2(40)	386 ms	*	*	11.2
		4(40)	1.2 s	*	*	8.6
		6(40)	2.5 s	*	*	6.3
Epin.	**30**	2(20)	4 s	*	*	11.5
		8(80)	29 s	*	*	13.8
	40	2(20)	10 s	*	*	14.6
		10(100)	69 mins	*	*	18
	50	2(20)	18 s	*	*	18.2
		10(100)	840 mins	*	*	19.8
		20(200)	*	*	*	23.9
	10	2(20)	22 ms	616	117	8
		2(30)	20 ms	600	159	9
		2(50)	20 ms	582	198	12.1
		5(50)	55 ms	982	462	8.8
	20	2(20)	43 ms	*	*	12.5
		2(40)	33 ms	1250	*	17
		4(40)	84 ms	*	*	13.6
		6(40)	158 ms	*	*	11.8
Amaz.	**30**	2(20)	88 ms	*	*	19.4
		8(80)	30 s	*	*	22.5
	40	2(20)	108 ms	*	*	23.3
		10(100)	70 mins	*	*	27.8
	50	2(20)	130 ms	*	*	28
		10(100)	870 mins	*	*	32.2
		20(200)	*	*	*	37.5

Table 5
Influence spread of the *Rand.(om)* and *CSA-Q* algorithms for two complex cases.

Method	**K**	**R** (η)	**NetH.**	**Wiki**	**Epin.**	**Amaz.**
Rand.	**40**	10(100)	90	75	165	158
	50	10(100)	315	283	334	351
CSA-Q	**40**	10(100)	112	93	196	192
	50	10(100)	377	343	402	456

We observe that the run time of *CSA-Q* decreases when *R* increases given fixed values of (K, η) (e.g. (20,40)). For a larger *R*, the *CSA-Q* algorithm is driven to search nodes with less influence since the failure of such nodes can still meet the constraint. The targeting nodes are normally with a small number of paths, which reduces the computational time (as analyzed in Section 4.4). For the *CSA* algorithm, the reduction is overwhelmed by the growth of run time needed to compute influence loss for more subsets as *R* increases.

Except the *Rand.(om)* and *CSA-Q* methods, the other two algorithms cannot solve most of the cases in all networks. Even the *Rand.(om)* method cannot solve the case ($K = 50$, $R = 20$) within one day. In contrast, the *CSA-Q* method can complete the search of top-*K* nodes in less than one hour for any of the networks. Instead of computing influence loss for every subset of the selected nodes, *CSA-Q* evaluates the constraint by summing individual influence, which is linear with the number of failure nodes. Hence it is scalable to solve all complex cases in our experiments within a reasonable amount of time.

To further confirm the quality of solutions produced by the *CSA-Q* algorithm, we show influence spread when the algorithm is compared to the *Rand.(om)* method - the other one can solve the complex cases within acceptable time (one day). Table 5 demonstrates that the *CSA-Q* algorithm achieves significantly larger influence spread while it costs much less time (as seen in Tables 3 and 4).

5.2.5. Summary

We show the impact of influence loss and reveal its importance in solving influence maximization problem. We further demonstrate that both the *CSA* and *CSA-Q* algorithms outperform the baseline methods, *GreedyB* and *Random*, in term of influence spread of top-*K* nodes. More importantly, the *CSA-Q* algorithm achieves significantly improvement on the scalability while maintaining sufficiently good solutions to the IMIL problem.

6. Conclusions

The IMIL problem is motivated by practical thoughts on viral marketing. We aim to find top-*K* influential nodes given influence loss constraint in social networks. This problem is proved to be NP-hardness and existing methods fail to provide reasonably good solutions. To solve the problem, we developed a CSA based framework that optimizes top-*K* solutions while enforcing satisfaction of influence loss constraint. The development of CSA algorithms is not trivial in the new problem context as we need to investigate algorithmic convergence according to a particular domain based penalty function and practical parameter settings. We further proposed an enhanced version of the CSA algorithm that employs a new penalty function, and showed its significant improvement on the algorithmic efficiency.

It is the first time that influence loss is considered in influence propagation in social networks. The proposed influence maximization technique is a reliable top-*K* solution to developing practical applications of social networks in a complex setting. Due to unpredictable factors, node failure may often occur in the real-world environment. For example, in knowledge diffusion networks, individuals may lose the propagation capability during the knowledge evolution (Luo, Du, Liu, Xuan, & Wang, 2015). Our technique may reduce the risk of knowledge loss in a knowledge-transfer process. This directly facilitates a robust development of expert systems on the knowledge elicitation and combination. On the other hand, the proposed CSA framework is shown to be very useful for solving other constrained optimization problems in social networks. We can perceive effective CSA based solutions to many optimization problems in the development of intelligent systems (Marinaki & Marinakis, 2016).

As a primitive step to investigate influence loss in social networks, we empirically study impact of the required inputs in the proposed technique: the number of failure nodes and the influence loss threshold. As demonstrated in Figs. 7 and 9, the two parameters exhibit expected impact in the influence propagation. A precise estimation on the parameter values will definitely direct the development of our technique on both its effectiveness and efficiency. Hence the proposed technique may require much effort from domain experts in the problem formulation and solution development. Meanwhile, we can observe that the CSA algorithm with the new penalty function still demands a large amount of time on solving very large networks since it needs to search the entire solution space. This may limit its real-time applications when the calculation shall be conducted online.

The previous limitations imply two lines of future research. On one hand, we can study behavior of failure nodes in social networks particularly in practical applications. The investigation may provide more insightful knowledge about influence loss in the real-time propagation, which in turn supplies exact inputs to the solution development. This study may simultaneously indicate potential strategies to avoid the node failure as well as to remedy the reduced influence when nodes fail in social networks. The strategies are valuable upon building reliable application systems and release a tedious task on eliciting domain knowledge for developing the solutions. On the other hand, we will continue to improve the CSA algorithm by pruning solutions with large influence loss in advance. We are particularly interested in examining the utility of domain knowledge on the CSA performance when a complex constrained optimization problem needs to be solved in applications.

Acknowledgment

The research was supported by the projects: NSFC 61375070, 61373033 andSZSTI JCYJ201418193546117.

References

Chen, W., Lu, W., & Zhang, N. (2012). Time-critical influence maximization in social networks with time-delayed diffusion process. In *Proceedings of the twenty-sixth aaai conference on artificial intelligence (aaai)* (pp. 592–598).

Chen, W., Wang, C., & Wang, Y. (2010). Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th acm sigkdd international conference on knowledge discovery and data mining (kdd)* (pp. 1029–1038).

Chen, W., Wang, Y., & Yang, S. (2009). Efficient influence maximization in social networks. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining (kdd)* (pp. 199–208).

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 (kdd)* (pp. 57–66).

Feng, S., Chen, X., Cong, G., Zeng, Y., Chee, Y. M., & Xiang, Y. (2014). Influence maximization with novelty decay in social networks. In *Proceedings of the twenty-eighth aaai conference on artificial intelligence (aaai)* (pp. 37–43).

Gomez-Rodriguez, M., & Scholkopf, B. (2012). Influence maximization in continuous time diffusion networks. In *Proceedings of the twenty-ninth international conference on machine learning (icml)*.

Goyal, A., Bonchi, F., Lakshmanan, L. V., & Venkatasubramanian, S. (2013). On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining, 3*(2), 179–192.

Hajian, B., & White, T. (2012). On measurement of influence in social networks. In *Asonam* (pp. 101–105).

Jacob, G., Barak, L., & Eitan, M. (2001). Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Marketing Letters, 12*(3), 211–223. doi:10.1023/A:1011122126881.

Jiang, Q., Song, G., Gao, C., Wang, Y., Si, W., & Xie, K. (2011). Simulated annealing based influence maximization in social networks. In *Proceedings of the twenty-fifth aaai conference on artificial intelligence (aaai)* (pp. 127–132).

Jung, K., Heo, W., & Chen, W. (2012). Irie: scalable and robust influence maximization in social networks. In *Proceedings of the 2012 ieee 12th international conference on data mining (icdm)* (pp. 918–923).

Kempe, D., Kleinberg, J., & Tardos, E. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the ninth acm sigkdd international conference on knowledge discovery and data mining (kdd)* (pp. 137–146).

Kim, J., Kim, S. K., & Yu, H. (2013). Scalable and parallelizable processing of influence maximization for large-scale social networks. In *Proceedings of the twenty-ninth international conference on data engineering (icde)* (pp. 266–277).

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 (kdd)* (pp. 420–429).

Li, G., Chen, S., Feng, J., Tan, K. l., & Li, W. S. (2014). Efficient location-aware influence maximization. In *Proceedings of the 2014 acm sigmod international conference on management of data (sigmod)* (pp. 87–98).

Liu, B., Cong, G., Xu, D., & Zeng, Y. (2012). Time constrained influence maximization in social networks. In *Proceedings of the 12th ieee international conference on data mining (icdm)* (pp. 439–448).

Liu, B., Cong, G., Zeng, Y., Xu, D., & Meng, C. Y. (2014a). Influence spreading path and its application to the time constrained social influence maximization problem and beyond. *IEEE Transactions on Knowledge and Data Engineering, 26*(8), 1904–1917.

Liu, X., Li, M., Li, S., Peng, S., Liao, X., & Lu, X. (2014b). Imgpu: Gpu-accelerated influence maximization in large-scale social networks. *IEEE Transactions On Parallel and Distributed Systems, 25*(1), 136–145.

Luo, S., Du, Y., Liu, P., Xuan, Z., & Wang, Y. (2015). A study on coevolutionary dynamics of knowledge diffusion and social network structure. *Expert Systems With Applications, 42*(7) 3619–3533.

Marinaki, M., & Marinakis, Y. (2016). A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. *Expert Systems With Applications, 46*(15), 145–163.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*(6), 1087–1092.

Nguyen, H., & Zheng, R. (2012). On budgeted influence maximization in social networks. arXiv:1204.4491v2.

Ostfeld, A., Uber, J. G., & Salomons, E. (2006). Battle of water sensor networks: a design challenge for engineers and algorithms. In *Proceedings of water distribution systems analysis conference*.

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 (kdd)* (pp. 61–70).

Song, G., Zhou, X., Wang, Y., & Xie, K. (2014). Influence maximization on large-scale mobile social network: a divide-and-conquer method. *IEEE Transactions On Parallel and Distributed Systems*, 1–14.

Sun, X., Lin, H., & Xu, K. (2015). A social network model driven by events and interests. *Expert Systems With Applications, 42*(9), 4229–4238.

Wah, B. W., & Wang, T. (1999). Simulated annealing with asymptotic convergence for nonlinear constrained global optimization. In *Principles and practice of constraint programming* (pp. 461–475). Springer-Verlag.

Wang, T. (2001). *Global optimization for constrained nonlinear programming*. University of Illinois at Urbana-Champaign Phd thesis.

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 (kdd)* (pp. 1039–1048).

Wasserman, S., & Faust, K. (1994). *Social network analysis: methods and applications*. Cambridge University Press.

Zafarani, R., Abbasi, M. A., & Liu, H. (2014). *Social media mining: an introduction*. Cambridge University Press.

Zeng, Y., Chen, X., Cao, X., Qin, S., Cavazza, M., & Xiang, Y. (2015). Optimal route search with the coverage of users' preferences. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence (ijcai)* (pp. 2118–2124).