Computational Social Networks
a SpringerOpen Journal

RESEARCH Open Access

Efficient influence spread estimation for influence maximization under the linear threshold model

Zaixin Lu*, Lidan Fan, Weili Wu, Bhavani Thuraisingham and Kai Yang

*Correspondence:
zaixinlu@utdallas.edu
Department of Computer Science,
University of Texas at Dallas, 800 W.
Campbell Road, Richardson, TX
75080, USA

Abstract

Background: This paper investigates the influence maximization (IM) problem in social networks under the linear threshold (LT) model. Kempe et al. (ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 137–146, 2003) showed that the standard greedy algorithm, which selects the node with the maximum marginal gain repeatedly, brings a $\frac{e-1}{e}$-factor approximation solution to this problem. However, Chen et al. (International Conference on Data Mining, pp. 88–97, 2010) proved that the problem of computing the expected influence spread (EIS) of a node is #P-hard. Therefore, to compute the marginal gain exactly is computational intractable.

Methods: We step-up on investigating efficient algorithm to compute EIS. We show that the EIS of a node can be computed by finding cycles through it, and we further develop an exact algorithm to compute EIS within a small number of hops and an approximation algorithm to estimate EIS without the hop constraint. Based on the proposed EIS algorithms, we finally develop an efficient greedy based algorithm for IM.

Results: We compare our algorithm with some well-known IM algorithms on four real-world social networks. The experimental results show that our algorithm is more accurate than others in finding the most influential nodes, and it is also better than or competitive with them in terms of running time.

Conclusions: IM is a big topic in social network analysis. In this paper, we investigate efficient influence spread estimation for IM under the LT model. We develop two influence spread estimation algorithms and a new greedy based algorithm for IM under the LT model. The performance of the proposed algorithms are analyzed theoretically and evaluated through simulations.

Keywords: Social network analysis; Expected influence spread estimation; Influence maximization; Linear threshold model

Background

Social network is a multidisciplinary research area for both academia and industry, including social network modeling, social network analysis, and data mining. An interesting problem in social network analysis is influence maximization (IM), which can be applied in marketing to deploy business strategies. Typically, IM is the problem that given a graph G as a social network, an influence spread model and an integer k select the top

k nodes as seeds to maximize the expected influence spread (EIS) through G. One corresponding issue in marketing is product promotion. In order to advertise a new product efficiently within a limited budget, a company may choose a few people as seeds who will be given free samples. It is likely that those people will recommend others, such as their friends, relatives or co-workers, to try this product. Eventually, a great number of people may adopt the product due to such 'word-of-mouth' effect [1-6]. Intuitively, the initial seed selection is a key factor that will impact on the success of the product promotion. Therefore, it is important to design applicative influence spread model and efficient search algorithm to find the most influential people in social networks.

IM was first investigated as an combinatorial optimization problem by Kempe et al. in [5]. They considered two influence spread models, namely, Independent Cascade (IC; [2,3]) and Linear Threshold (LT; [7,8]), and proved a series of theoretical results. After that, the two models have been extensively studied (please see, e.g., [9-15] for recent works). In this paper, we focus upon the LT model. Let S be a set of initially active nodes; the influence, under the LT model, propagates in a threshold manner. That is, a node v is activated if and only if the sum of influence it receives from its active neighbors exceeds a threshold $\lambda(v)$ chosen uniformly at random.

As we understand, a crucial part of IM is how to compute the EIS given a node, since only we know the EIS of each node, and then we could find a seed set to maximize the combinatorial EIS. The exact EIS computation was left as an open problem in [5] and has attracted a great deal of attentions in recent years (see, e.g., [9-11,13,15,16]). In [11], Chen et al. proved that computing the exact EIS under the LT model is #P-hard. Therefore, a polynomial time exact solution does not exist unless $P = \text{NP}$. But based on the observations in [11,15], the influence diminishes rapidly during the diffusion in many real-world social networks under the LT model. In other words, the influence spread of a seed is limited within a small number of hops. It has been shown that the influence spread under the LT model can be computed by searching simple paths starting from the seeds [11,15]. Therefore, we can define a hop constraint T such that given a seed v, we only take paths within T hops to estimate the EIS of v. The main contributions of this paper are as follows:

1. We develop an exact algorithm for computing the EIS within four hops. Instead of finding simple paths, we compute the EIS of a node by finding cycles through it. In this study, a cycle of length l is defined as a path visiting a node twice and visiting other $l - 2$ nodes exactly once. The detailed algorithm is given in the 'Methods' section.
2. For the case that $T > 4$, we develop an approximation algorithm to estimate EIS based on random walk. The experimental results in the 'Results and discussion' section show that more precise and quick results can be obtained by using a combination of our exact and approximation algorithms rather than using methods based on simple path.
3. When applying the standard greedy algorithm to IM, it will repeatedly run EIS estimation (EISE) until the top k influential nodes are selected. To further reduce the running time, we construct two lists to save the influence diffused by each node and the active probability of each node, respectively. Moreover, we develop two algorithms to update the two lists when adding a new seed so that the next one with the maximum marginal gain can be directly obtained without running the EISE.

The update algorithms are represented in the 'Influence maximization' section. It is able to say that the two lists contain all the information for doing the seed selection, and they can be easily and quickly updated by our update algorithms.

4. We compare our algorithm with some well-known IM algorithms on four real-world social networks. The experimental results show that our algorithm is more accurate than others in finding the most influential nodes, and it is also better than or competitive with them in terms of running time.

The rest of this paper is organized as follows: The 'Related work' section introduces the related works. 'Problem description' section gives the problem descriptions of both EISE and IM. 'Methods' and 'Influence maximization' sections study the two problems, respectively. In detail, 'A deterministic algorithm' section efficiently solves the EISE assuming that the influence spread is negligible after four hops. 'A randomized algorithm' section presents an approximation algorithm for general EISE. The 'Influence maximization' section presents a fast method to solve IM by using the algorithms proposed in the 'Methods' section. Finally, 'Results and discussion' section gives the simulation results, and the 'Conclusion' section concludes this paper.

Related work

In the literature, the IM problem has been extensively studied under the IC and LT models. Kempe et al. in [5] first showed that it is NP-hard to determine the optimum for IM under the two models, and by showing that the EIS function is monotone and submodular, they proved that the standard greedy algorithm brings a $\frac{e-1}{e}$-factor approximation solution. In mathematics, a set function $f : 2^{\Omega} = \mathbb{R}^{+}$ is monotone and submodular if $\forall S_2 \subseteq S_1$, we have $f(S_1) \geq f(S_2)$ and $f(S_1 \cup \{u\}) - f(S_1) \geq f(S_2 \cup \{u\}) - f(S_2)$, where u is an arbitrary item. In such cases, a $\frac{e-1}{e}$-factor approximation solution can be obtained by picking the item with the maximum marginal gain repeatedly [17]. In [5], how to compute the exact marginal gain (i.e., compute the EIS increment when adding a node) under the two models was left as an open problem, and they estimated it by running the Monte Carlo (MC) simulation, which is not computational efficient (e.g., it takes days to select 50 seeds in a moderate size graph of 30K nodes [11]). Motivated by improving the running time performance, many algorithms have been proposed. Leskovec et al. developed a Cost-Effective Lazy Forward (CELF) algorithm, which is up to 700 times faster than the greedy algorithm with Monte Carlo simulation [16]. But as the results shown in [9], CELF still cannot be applied to find seeds in large social networks, and it takes several hours to select 50 seeds in a graph with tens of thousands of nodes. To further reduce the running time, Goyal et al. [13] developed an extension of CELF, called CELF++, which was showed 0.35 to 0.55 faster than CELF. In [9], Chen et al. proposed two new greedy algorithms, namely NewGreedy and MixedGreedy. NewGreedy reduces the running time by deleting edges having no contribution to influence spread (similar idea was also proposed in [18]), and MixedGreedy which is a combination of NewGreedy and CELF (it uses NewGreedy as the first step and applies CELF for the remaining rounds). Based on the experiments, they showed that MixedGreedy is much faster than both NewGreedy and CELF.

Based on the IC model, Chen et al. also proposed a new influence spread model, called Maximum Influence Arborescence (MIA), to further reduce the running time of EISE.

The efficiency of MIA was demonstrated in [10]. Besides selecting nodes greedily, Wang et al. [19] proposed a community-based algorithm for mining the top k influential nodes under the IC model, and Jiang et al. in [14] proposed a heuristic algorithm based on Simulated Annealing.

In terms of LT model, after Kempe et al. proposed the greedy algorithm [5], the most recent works for IM under this model are [10,12,15]. In [10], Chen et al. proved that the EIS under LT model can be computed in linear time in a directed acyclic graph, and they proposed an algorithm called Local Directed Acyclic Graph (LDAG). Given a general graph, it first converts the original graph into small acyclic graphs, and it only considers the EIS of a node within its local graph when computing the marginal gain. In [12], Narayanam and Narahari developed an algorithm for the LT model that selects the nodes based on the Shapley Value. In [15], Goyal et al. proposed an algorithm called SIMPATH, which estimates the EIS by searching for the simple paths starting from seeds. Since it is computationally expensive to find all the simple paths, they adopted a parameter η to prune them. They also applied the vertex cover optimization to cut down the number of iterations. Based on their experimental results, SIMPATH showed its merits from the aspects of running time and seed quality.

Problem description

Many introductions about the LT model and IM problem can be found in detail in papers cited above. Here, for the sake of completeness, we give a brief description for the LT model and formal definitions for IM and EISE.

Definition 1. Let $G(V, E)$ be a directed graph; we define

- $N_{\text{in}}(v)$ (respectively $N_{\text{out}}(v)$) to be the set of incoming (respectively outgoing) neighbors of v ($\forall v \in V$).
- $\lambda(v)$ to be the threshold of v, which is a real number in the range of $[0, 1]$ chosen uniformly at random.
- $x(v)$ to be a 0 to 1 variable which indicates whether v is active or not.

According to Definition 1, given a weighted directed graph $G(V, E, w)$, where $w(e) \in [0, 1]$ ($\forall e \in E$) is a weight function, the sum of influence v receives can be formulated as $\sum_{u \in N_{\text{in}}(v)} x(u)w(u, v)$. Without loss of generality, we assume $\sum_{u \in N_{\text{in}}(v)} w(u, v) \leq 1$ ($\forall v \in V$). In the LT model, time is discrete. Given a seed set S, at time 0, we have $\forall v \in S$, $x(v) = 1$, and $\forall u \in (V \backslash S)$, $x(u) = 0$. At any particular time t, a node $v \in V$ becomes active if $\sum_{u \in N_{\text{in}}(v)} x(u)w(u, v) \geq \lambda(v)$. Finally, the influence spread process stops at a time slot when there is no newly activated node.

Definition 2. *EISE*: Given a weighted directed graph $G(V, E, w)$ and a set $S \subseteq V$ of nodes, EISE is the problem of estimating the expected number of active nodes at the end of the influence spread. EISE$_T$ is the problem that given an integer T, estimates the expected number of nodes that are active at time T.

For the rest of this paper, given a seed set S, we denote by $\sigma(S)$ the expected number of nodes that are eventually active and denote by $\sigma_T(S)$ the expected number of nodes that are activated within T time slots. We can say that $\sigma(S)$ is an expected number among the probability distributions of active nodes given S and $\sigma_T(S)$ is a time limited version of $\sigma(S)$.

Definition 3. *IM:* Given a weighted directed graph $G(V, E, w)$ and a parameter k, the IM problem is to find a seed set S of cardinality k to maximize $\sigma(S)$.

As the experimental results shown in [15], under the LT model, the EIS is negligible after a small number of hops (usually three or four hops) in many real-world social networks. Therefore, to solve the IM problem, it is sufficient to compute $\sigma_T(S)$ instead of $\sigma(S)$ for some small value of T.

Methods

We first present a deterministic algorithm for computing the exact value of $\sigma_T(v)$ for the case that $T \leq 4$ in the 'A deterministic algorithm' section and then present a randomized algorithm for estimating $\sigma_T(v)$ for $T \geq 5$ in the 'A randomized algorithm' section.

Definition 4. In this study, we define

- a path is a sequence of nodes, each of which is connected to the next one in the sequence; and a path with no repeated nodes is called a simple path.
- a cycle is a path such that the first node appears twice and the other nodes appear exactly once; and a simple cycle is a cycle such that the first and last nodes are the same.

A deterministic algorithm

According to the observation in [15], the EIS of a node v after three or four hops is negligible in most cases. Therefore, we are interested in how to compute $\sigma_T(v)$ for $T \leq 4$. In [11], it has been shown that the EIS of a seed set S under the LT model can be formulated as

$$\sigma(S) = \sum_{\pi \in \mathcal{P}(S)} \prod_{e \in \pi} w(e) + |S| \quad [15],$$

where $\mathcal{P}(S)$ denotes the set of simple paths starting from nodes in S, π denotes an element in $\mathcal{P}(S)$, and e denotes an edge in π. Thus, $\forall v \in V$, we have

$$\sigma(v) = \sum_{\pi \in \mathcal{P}(v)} \prod_{e \in \pi} w(e) + 1,$$

where $\mathcal{P}(v)$ denotes the set of simple paths starting from node v.

As an example shown in Figure 1, considering v_0 is an active node, then the probability that v_4 can be activated by v_0 is $w(0,1)w(1,4) + w(0,2)w(2,4) + w(0,3)w(3,4)$, which is

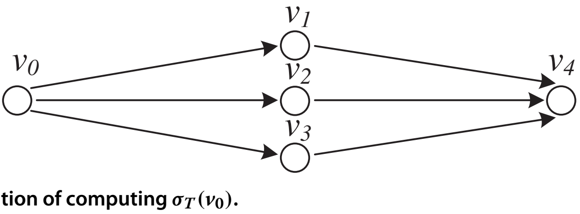


Figure 1 An illustration of computing $\sigma_T(v_0)$.

the sum of weight products of all the simple paths from v_0 to v_4. Although the example is easy to understand, in a general graph G, it requires exponential time to enumerate all the simple paths. Thus, to compute the exact value of $\sigma(v)$ is computational intractable, and a hop constraint T is used in this paper to balance the accuracy of EISE and the program efficiency in terms of running time.

In order to find a node v with the maximum $\sigma_T(v)$, we have to compute $\sigma_T(v)$ for all the nodes $v \in V$. Let $\sigma_0(v) = 1$ $(\forall v \in V)$; we first consider the simple case that $T = 1$. In such cases, we have $\sigma_1(v) = \sigma_0(v) + \sum_{u \in N_{\text{out}}(v)} w(v,u)$, because there is only direct influence spread without propagation. When $T > 1$, we can compute $\sigma_T(v)$ by recursively finding all the simple paths of length no more than T, starting from v, which requires $O(\Delta^T)$ time by using the depth-first search (DFS) algorithm, and Δ denotes the node maximum degree. Thus, let G be a weighted directed graph; computing $\sigma_T(v)$ for all the nodes in G requires $O(n\Delta^T)$ time if we use the above simple path method [15], where n denotes the number of nodes in G. To further improve the running time performance, we develop a dynamic programming (DP) approach to compute $\sigma_T(v)$ for $T \leq 4$. It is based on searching cycles instead of simple paths.

As an example shown in Figure 2, there are three types of cycles of length 4, and only the third one is a simple cycle. Let $\mathcal{C}_l(v)$ denote the set of cycles of length l, starting from v, and let

$$\varrho_T(v) = \sum_{l=2\cdots T} \sum_{\pi \in \mathcal{C}_l(v)} \prod_{e \in \pi} w(e),$$

we have

$$\begin{aligned}\sigma_T(v) &= \sigma_0(v) + \sum_{u \in N_{\text{out}}(v)} w(v,u) \cdot (\sigma_{T-1}^{V\setminus v}(u)) \\ &= \sigma_0(v) + \sum_{u \in N_{\text{out}}(v)} w(v,u) \cdot (\sigma_{T-1}(u)) \\ &\quad -\varrho_T(v),\end{aligned}$$

where $\sigma_{T-1}^{V\setminus v}(u)$ denotes the EIS of node u in the induced graph of $V\setminus v$ within $T-1$ hops, and $\varrho_T(v)$ denotes the invalid influence spread involving cycles.

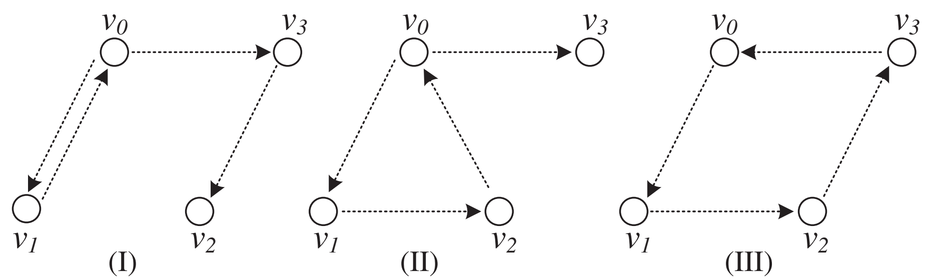


Figure 2 Three cases of cycles with length 4.

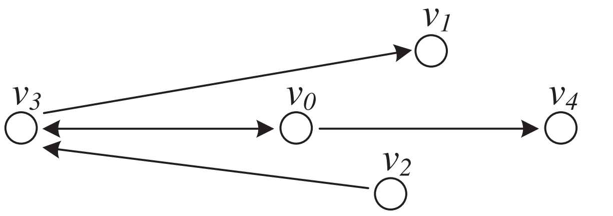


Figure 3 An illustration of computing $\varrho_3(v_0)$.

Figure 3 shows an example, in which v_3 and v_4 are v_0's outgoing neighbors. It is easy to see $\sigma_2(v_3) = 1 + w(3,0) + w(3,1) + w(3,0)w(0,4)$ and $\sigma_2(v_4) = 1$. Thus,

$$\begin{aligned}&\sigma_0(v_0) + \sum_{u \in N_{\text{out}}(v_0)} w(v_0,u) \cdot (\sigma_2(u)) \\ &= w(0,3) + w(0,4) + w(0,3)w(3,0) + w(0,3)w(3,1) \\ &\quad + w(0,3)w(3,0)w(0,4) + 1,\end{aligned}$$

in which the terms $w(0,3)w(3,0)$ and $w(0,3)w(3,0)w(0,4)$ have to be removed since they involve cycles. The rest of this section is devoted to investigating how to compute $\varrho_T(v)$ for $T \leq 4$.

Lemma 1. *Given a weighted directed graph $G(V,E,w)$ and an arbitrary node $v \in V$, $\varrho_T(v)$ can be computed in $O\left(\Delta^2\right)$ time when $T \leq 4$.*

A brief description for the idea of our method is presented before the formal algorithm and its proof. Firstly, $\varrho_T(v)$ involves all the cycles of length no more than T, starting from v. In order to compute $\varrho_4(v)$ efficiently, we divide $\varrho_4(v)$ into three parts: $\sum_{\pi \in \mathcal{C}_l(v)} \prod_{e \in \pi} w(e)$ $(l = 2,3,4)$ to carry on the analysis. If each part can be computed in $O(\Delta^2)$ time, $\varrho_4(v)$, which is the sum of them, can be obtained in $O(\Delta^2)$ time. Secondly, considering $\mathcal{C}_l(v)$ $(2 \leq l \leq 4)$, we can further classify the cycles in $\mathcal{C}_l(v)$ into $l-1$ types. Note that a cycle of length l, starting from v, consists of a sequence of $l+1$ nodes, two of which are v and others are distinct. Therefore, we can label a cycle according to the position in the sequence where the second v appears. $\forall v \in V$, let $\mathcal{C}_T^l(v)$ denote the set of cycles of length T, whose lth node is v, we have

$$\begin{aligned}\varrho_T(v) &= \sum_{l=2,\cdots,T} \sum_{\pi \in \mathcal{C}_l(v)} \prod_{e \in \pi} w(e) \\ &= \sum_{l=2,\cdots,T} \sum_{l'=3,\cdots,l+1} \sum_{\pi \in \mathcal{C}_l^{l'}(v)} \prod_{e \in \pi} w(e).\end{aligned}$$

In order to compute $\varrho_T(v)$, our method will compute each $\sum_{\pi \in \mathcal{C}_l^{l'}(v)} \prod_{e \in \pi} w(e)$ separately.

Proof. We will prove Lemma 1 by showing that $\sum_{\pi \in \mathcal{C}_l(v)} \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time when $l = 4$, and for the case that $l < 4$, $\sum_{\pi \in \mathcal{C}_l(v)} \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time or less via a similar method. As we have mentioned above, there are only three types of cycles of length 4, as shown in Figure 2.

Consider case (I). Such a cycle consists of a simple cycle of length 2 and a simple path of length 2. Let $\mathcal{P}_2(v)$ denote the set of simple paths of length 2, starting from v, and $\mathcal{C}_2^3(v)$ denote the set of simple cycles of length 2 through v. $\mathcal{P}_2(v)$ can be obtained in $O(\Delta^2)$ time by DFS, and $\mathcal{C}_2^3(v)$ can be obtained by finding the set of nodes that are both incoming and outgoing neighbors of v, i.e.,

$$\mathcal{C}_2^3(v) = \left\{(v, u, v) : u \in N_{\text{out}}(v) \cap N_{\text{in}}(v)\right\}.$$

The intersection of two lists can be obtained in linear time if the two lists are sorted. Let $I(v) = N_{\text{out}}(v) \cap N_{\text{in}}(v)$ and $\kappa = \sum_{u \in I(v)} w(v, u)w(u, v)$; we have

$$\begin{aligned}
&\sum_{\pi \in \mathcal{C}_4^3(v)} \prod_{e \in \pi} w(e) \\
&= \sum_{\pi \in \mathcal{P}_2(v)} \sum_{u \in I(v) \backslash \pi} w(v, u)w(u, v) \prod_{e \in \pi} w(e) \\
&= \sum_{\pi \in \mathcal{P}_2(v)} \left(\kappa - \sum_{u \in \pi \cap I(v)} w(v, u)w(u, v)\right) \prod_{e \in \pi} w(e) \\
&= \sum_{\pi \in \mathcal{P}_2(v)} \left(\kappa - \sum_{u \in \pi \backslash v} w(v, u)w(u, v)\right) \prod_{e \in \pi} w(e),
\end{aligned}$$

in which $I(v)\backslash\pi$ denotes the set of nodes in $I(v)$ but not in π, $e \in \pi$ denotes an edge in π, and $u \in \pi$ denotes a node in π. Note that if $u \notin I(v)$, we have $(v, u) \notin E$ or $(u, v) \notin E$. In such cases, $w(v, u)w(u, v) = 0$. Therefore, $\sum_{u \in \pi \cap I(v)} w(v, u)w(u, v) = \sum_{u \in \pi \backslash v} w(v, u)w(u, v)$. Since $\mathcal{P}_2(v)$ consists of at most Δ^2 elements, each of which includes only two edges, $\sum_{\pi \in \mathcal{P}_2(v)}(\kappa - \sum_{u \in \pi \backslash v} w(v, u)w(u, v)) \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time.

Consider case (II). $\sum_{\pi \in \mathcal{C}_4^4(v)} \prod_{e \in \pi} w(e)$ can be computed by a similar method. A cycle in $\mathcal{C}_4^4(v)$ consists of a simple cycle of length 3, in which the first and last nodes are v. Therefore, instead of directly constructing a set of simple cycles of length 3, we can construct a set $\mathcal{P}_2(v)$ of simple paths of length 2. Let $l(\pi)$ denote the last node of a path $\pi \in \mathcal{P}_2(v)$ and let $\tau = \sum_{u \in N_{\text{out}}(v)} w(v, u)$; we have

$$\begin{aligned}
&\sum_{\pi \in \mathcal{C}_4^4(v)} \prod_{e \in \pi} w(e) \\
&= \sum_{\pi \in \mathcal{P}_2(v)} w(l(\pi), v) \left(\sum_{u \in N_{\text{out}}(v) \backslash \pi} w(v, u)\right) \prod_{e \in \pi} w(e) \\
&= \sum_{\pi \in \mathcal{P}_2(v)} w(l(\pi), v) \left(\tau - \sum_{u \in \pi \backslash v} w(v, u)\right) \prod_{e \in \pi} w(e),
\end{aligned}$$

in which $w(l(\pi), v) = 0$ if $l(\pi) \notin N_{\text{in}}(v)$. Therefore, $\sum_{\pi \in \mathcal{C}_4^4(v)} \prod_{e \in \pi} w(e)$ can also be computed in $O(\Delta^2)$ time.

Consider case (III). The analysis is somewhat more complicated. Instead of computing $\sum_{\pi \in \mathcal{C}_4^5(v)} \prod_{e \in \pi} w(e)$ directly, we first show that $\sum_{\pi \in (\mathcal{C}_4^5(v) \cup \mathcal{C}'(v))} \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time, where $\mathcal{C}'(v)$ denotes the set of cycles as shown in Figure 4. That is, cycles consist of three nodes in which the first two nodes are visited twice. Let $\rho_2(v, v')$ denote the probability that v' is reachable from v with exact two hops, i.e.,

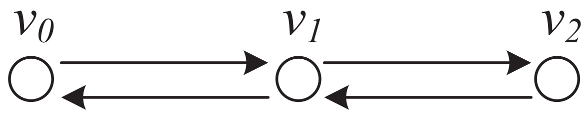


Figure 4 An invalid case.

$\rho_2(v, v') = \sum_{u \in N_{\text{out}}(v) \cap N_{in}(v')} w(v, u)w(u, v')$. Let $N^2_{\text{out}}(v)$ be the set of nodes that are reachable from v with exact two hops. To compute $\rho_2(v, v')$ for all the nodes $v' \in N^2_{\text{out}}(v)$, we can build up an outgoing tree rooted at v, in which the nodes are repeatable among different paths. This can be done in $O(\Delta^2)$ time by DFS. In addition, let $N^2_{\text{in}}(v)$ be the set of nodes that can reach v with exact two hops, we can build up an incoming tree rooted at v to compute $\rho_2(v', v)$ for all the nodes $v' \in N^2_{\text{in}}(v)$ in the same way. Then, we have

$$\sum_{\pi \in \left(\mathcal{C}^5_4(v) \cup \mathcal{C}'(v)\right)} \prod_{e \in \pi} w(e)$$

$$= \sum_{v' \in N^2_{\text{out}}(v) \cap N^2_{\text{in}}(v)} \rho_2(v, v')\rho_2(v', v),$$

which can be computed in $O(\Delta^2)$ time. It is easy to see

$$\sum_{\pi \in \mathcal{C}^5_4(v)} \prod_{e \in \pi} w(e)$$

$$= \sum_{\pi \in \left(\mathcal{C}^5_4(v) \cup \mathcal{C}'(v)\right)} \prod_{e \in \pi} w(e) - \sum_{\pi \in \mathcal{C}'(v)} \prod_{e \in \pi} w(e).$$

Therefore, to show $\sum_{\pi \in \mathcal{C}^5_4(v)} \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time, it is sufficient to show that $\sum_{\pi \in \mathcal{C}'(v)} \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time. We have

$$\sum_{\pi \in \mathcal{C}'(v)} \prod_{e \in \pi} w(e)$$

$$= \sum_{v' \in I(v)} \sum_{u \in I(v')} w(v, v')w(v', u)w(u, v')w(v', v),$$

where $I(v) = N_{\text{out}}(v) \cap N_{\text{in}}(v)$ and $I(v') = N_{\text{out}}(v') \cap N_{\text{in}}(v')$. Therefore, $\sum_{\pi \in \mathcal{C}'(v)} \prod_{e \in \pi} w(e)$ can be computed in $O(\Delta^2)$ time.

In sum, we prove $\sum_{\pi \in \mathcal{C}_4(v)} \prod_{e \in \pi} w(e)$ ($\forall v \in V$) can be computed in $O(\Delta^2)$ time. It can be shown that $\sum_{\pi \in \mathcal{C}_l(v)} \prod_{e \in \pi} w(e)$ ($l < 4$) can be computed in $O(\Delta^2)$ time or less by a similar method. Therefore, it requires only $O(\Delta^2)$ time to compute $\varrho_4(v)$ ($\forall v \in V$). □

Theorem 1. *Given a weighted directed graph $G(V, E, w)$, Algorithm 1 can compute $\sigma_4(v)$ for all the nodes $v \in V$ in $O(n\Delta^2)$ time, where n denotes the number of nodes in V, and Δ denotes the maximum node degree.*

Algorithm 1 EISE$_4$

```
input: a weighted directed graph $G = (V, E, w)$.
let $\sigma_1(v) = \sum_{u \in N_{out}(v)} w_{v,u}$ $(\forall v \in V)$;
for $l = 2 \cdots 4$ do
    $\sigma_l(v) = \sigma_1(v) - \varrho_l(v) + \sum_{u \in N_{out}(v)} w_{v,u} \cdot \sigma_{l-1}(u)$;
end for
output: a list of $\sigma_4(v)$ for all the nodes $v \in V$.
```

Proof. Without considering the possible numerical computation error, the solution of Algorithm 1 is exact, and the time complexity analysis easily follows the algorithm. The computation of $\sigma_l(v)$ only depends on $\sigma_{l-1}(u)$ $(u \in N_{\text{out}}(v))$ and $\varrho_l(v)$. Therefore, $\sigma_4(v)$ for all the nodes $v \in V$ can be computed by a DP approach. The number of subproblems is $O(n)$ and each subproblem can be solved in $O(\Delta^2)$ time. Therefore, Algorithm 1 requires $O(n\Delta^2)$ time. □

Compared with the method based on a simple path, which requires $O(\Delta^4)$ time to compute $\sigma_4(v)$ for a node v, the core advantage of Algorithm 1 is its running time performance. Based on our experiments in the 'Results and discussion' section, when $T \leq 4$, Algorithm 1 can compute the $\sigma_T(v)$ for all the nodes in a moderate size graph in about 1 s.

A randomized algorithm

Theorem 1 shows that Algorithm 1 can efficiently compute $\sigma(v)$, if the EIS from node v is negligible after four hops. For the case that the EIS within a large number T hops is not negligible, it has been shown that computing $\sigma_T(v)$ is #P-hard [11]. To estimate $\sigma_T(v)$ approximately, we can use MC simulation, i.e., simulate the influence spread process a sufficient number of times, re-choosing the thresholds uniformly at random, and use the arithmetic mean of the results instead of the EIS. Let $X_1, X_2, \cdots, X_r$ be the numbers of active nodes at time T for r runs, and let $E[\overline{X}]$ be the EIS within time T. By Hoeffding's inequality [20], we have

$$\Pr\left(|\overline{X} - \mathrm{E}[\overline{X}]| \geq \epsilon\right) \leq \exp\left(-\frac{2\epsilon^2 r^2}{\sum_{i=1}^{r}(b_i - a_i)^2}\right),$$

where a_i and b_i are the lower and upper bounds for X_i, respectively. Apparently, $a_i \geq 0$ and $b_i \leq n$, where n is the number of nodes in the graph. Thus, $\forall 0 < \delta < 1$, when $r \geq \frac{n \ln \frac{1}{\delta}}{2\epsilon^2}$, the probability that $|\overline{X} - \mathrm{E}[\overline{X}]| \geq \epsilon$ is at most δ. Therefore, the EIS estimated by using MC simulation with a sufficient number of runs is nearly exact. However, as the experiments shown in [5,11,15], applying the MC simulation to estimate the EIS is computational expensive, and the standard greedy algorithm with MC simulation (run 10,000 times to get the average) requires days to select 50 seeds in some real-world social networks with tens of thousands of nodes.

To improve the computation efficiency, we developed a randomized algorithm, computing $\sigma_T(v)$ for $T \geq 5$. We first give the main idea of our method. Recall that the EIS of a node v can be computed by searching simple paths starting from v; thus, $\sigma_T(v) = \sum_{\pi \in \mathcal{P}_T(v)} \prod_{e \in \pi} w(e)$. Let $\text{avg}(\mathcal{P}_T(v))$ be the arithmetic mean of $\prod_{e \in \pi} w(e)$ for all the elements $\pi \in \mathcal{P}_T(v)$, and let $|\cdot|$ be the number of elements in '$\cdot$'; we have $\sigma_T(v) = \text{avg}(\mathcal{P}_T(v))|\mathcal{P}_T(v)|$. However, obtaining $\text{avg}(\mathcal{P}_T(v))$ and $|\mathcal{P}_T(v)|$ requires the knowledge of $\mathcal{P}_T(v)$ and is therefore as difficult as the original problem. We propose an alternative approach. Instead of computing $\sigma_T(v)$ directly, we relax $\mathcal{P}_T(v)$ to $\acute{\mathcal{P}}_T(v)$ that contains all the paths starting from v, instead of simple paths. Let $x(\pi \in \mathcal{P}_T(v))$ be a 0 to 1 variable denote whether π is a simple path or not; we have

$$\sigma_T(v) = \sum_{\pi \in \acute{\mathcal{P}}_T(v)} x\big(\pi \in \mathcal{P}_T(v)\big) \prod_{e \in \pi} w(e).$$

The next question is how to estimate $\text{avg}(\acute{\mathcal{P}}_T(v))$ and $|\acute{\mathcal{P}}_T(v)|$ to obtain $\sigma_T(v)$.

Lemma 2. *Given a directed graph $G(V, E)$ and an integer T, there is a polynomial time algorithm to compute $|\acute{\mathcal{P}}_T(v)|$ for all the nodes $v \in V$.*

Proof. We can compute $|\acute{\mathcal{P}}_T(v)|$ by iteration or recursion. $\forall 1 \leq l \leq T$, we have

$$\begin{cases} |\acute{\mathcal{P}}_l(v)| = |\mathcal{P}_l(v)| = |N_{\text{out}}(v)|, & l = 1 \\ |\acute{\mathcal{P}}_l(v)| = \sum_{u \in N_{\text{out}}(v)} |\acute{\mathcal{P}}_{l-1}(v)|, & \text{otherwise.} \end{cases}$$

$|\acute{\mathcal{P}}_1(v)|$ equals to the number of outgoing neighbors of v, and $|\acute{\mathcal{P}}_l(v)|$ $(l > 1)$ can be obtained by a DP approach. Since there are $O(nT)$ subproblems and each subproblem can be solved in $O(\Delta)$ time, $|\acute{\mathcal{P}}_T(v)|$ can be obtained in $O(nT\Delta)$ time. □

Theorem 2. *Let ϵ and δ be two positive constants in the range of $(0, 1)$. There is a random walk algorithm such that given a weighted directed graph $G(V, E, w)$ and a node $v \in V$, it gives a $(1 \pm \epsilon)$-factor approximation solution to avg$(\acute{\mathcal{P}}_T(v))$ in $O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta} + nT\Delta\right)$ time with probability greater than $1 - \delta$.*

Proof. We can use uniform random sampling, which selects elements with equal probability from $\acute{\mathcal{P}}_T(v)$. By Lemma 2, we can obtain $|\acute{\mathcal{P}}_T(v)|$ for all the nodes $v \in V$ in $O(nT\Delta)$ time. Let the probability $\Pr(y_{i+1} = u'|y_i = u) = \frac{|\acute{\mathcal{P}}_{T-i}(u')|}{|\acute{\mathcal{P}}_{T-i+1}(u)|}$ and $\Pr(y_1 = v) = 1$; then, a path of length T can be generated by taking T successive random steps. $\forall$ a path $\pi = (v_1, v_2, \cdots, v_T)$ in $\acute{\mathcal{P}}_T(v)$, we have

$$\begin{aligned} \Pr(\pi) &= \prod_{i=1,\cdots,T-1} \Pr(y_{i+1} = v_{i+1}|y_i = v_i) \\ &= \prod_{i=1,\cdots,T} \frac{|\acute{\mathcal{P}}_{T-i}(v_{i+1})|}{|\acute{\mathcal{P}}_{T-i+1}(v_i)|} = \frac{1}{|\acute{\mathcal{P}}_T(v)|}. \end{aligned}$$

Therefore, we can generate paths $\pi_1, \pi_2, \cdots, \pi_r$ uniformly at random. By Hoeffding's inequality, we have

$$\begin{aligned} &\Pr\left(\frac{|\sum_{i=1,\cdots,r} \prod_{e \in \pi_i} w(e) r}{-} \text{avg}\left(\acute{\mathcal{P}}_T(v)\right) | \geq \epsilon\right) \\ &\leq \exp\left(-\frac{2\epsilon^2 r^2}{\sum_{i=1}^{r} \left(\max_{\pi \in \acute{\mathcal{P}}_T(v)} \prod_{e \in \pi} w(e)\right)^2}\right), \end{aligned}$$

where $\max_{\pi\in\acute{\mathcal{P}}_T(v)} \prod_{e\pi} w(e)$ is the maximum weight product of a path of length T starting from v. Since $w(e) \leq 1$ ($\forall e \in E$), we have $\max_{\pi\in\acute{\mathcal{P}}_T(v)} \prod_{e\pi} w(e) \leq 1$. Thus, Theorem 2 is proved. □

Based on Theorem 2, we now describe our randomized algorithm for computing $\sigma_T(v)$ for all the nodes $v \in V$. It runs in $O(nT\Delta + nr)$ time, where r is a constant and does not depend on the input graph.

In Algorithm 2, it first computes $|\acute{\mathcal{P}}_T(v)|$ (step 1) and then estimates $\sigma_T(v)$ by uniform random sampling. As far as the running time, the most time-consuming part is steps 2 to 8, in which r is independent of the input graph. It is clear that when r is small, the accuracy of EISE is low, but the estimation time is short, and vice verse. Compared with MC simulation, Algorithm 2 is much faster. In order to estimate the EIS of a node, it only generates a constant number of paths, while if MC simulation is applied instead of Algorithm 2, each time we have to re-choose the thresholds for all the nodes, and the time complexity is $O((|V| + |E|)r)$, when most of the edges are accessed each time. In the experiment, we observed that the error is less than 3% when $T = 5$, using an appropriate number of samples ($r = 1,000$).

Algorithm 2 EISE$_T$

```
0:  input: a weighted directed graph $G = (V, E, w)$ and two integers $T$ and $r$.
1:  construct $|\acute{\mathcal{P}}_l(v)|$ $(1 \leq l \leq T)$ for all the nodes $v \in V$;
2:  for $v \in V$ do
3:      let $\sigma_T(v) = 0$;
4:      for $i = 1, \cdots, r$ do
5:          let $\pi_r$ be the path of length $T$, generated by the random walk technique;
6:          $\sigma_T = \sigma_T + x(\pi_r \in \mathcal{P}_T(v)) \prod_{e\in\pi_r} w(e)$;
7:      end for
8:  end for
9:  $\sigma_T(v) = \sigma_T(v)\frac{|\acute{\mathcal{P}}_l(v)|}{r}$;
10: output: a list of $\sigma_T(v)$ for all the nodes $v \in V$.
```

Influence maximization

Considering the computational efficiency, we define a hop constraint for EISE, and we present two algorithms in 'Methods' section to compute $\sigma_T(v)$ in v's local area (T hops). The proposed algorithms are worth applying to solve the IM problem greedily. Given a weighted directed graph $G(V, E, w)$, the standard greedy algorithm will run EISE $O(n)$ times to select a seed, where n denotes the number of nodes. To further reduce the running time, we construct an influence list IL to store the EIS of nodes in the induced graph of $G\backslash S$, where S is the current seed set. Let $v_1, v_2, \cdots, v_n$ be the nodes in the input graph. Given a parameter T, initially we have IL $= \{l_1 = \sigma_T(v_1), \cdots, l_n = \sigma_T(v_n)\}$, since $S = \emptyset$. After adding a node v_i into S, all the nodes, whose local area include v_i, have to be updated. Instead of running EISE, we update them by building an incoming tree rooted at v_i (Algorithm 3).

Algorithm 3 Update$_{IL}$

```
0: input: $G = (V, E, w)$, $v$, $S$, and IL.
1: construct an incoming tree of depth $T$ rooted at $v$ in the induced graph of $G \backslash S$
   (without loss of generality, assume that the simple paths are $\pi_1, \pi_2, \cdots, \pi_m$);
2: for $i = 1 \cdots m$ do
3:     let $i_0, i_1, \cdots, i_T$ be the nodes visited by $\pi_i$ sequentially and $l_{i_0}, l_{i_1}, \cdots, l_{i_T}$ be the
       corresponding elements in IL (in which $i_0 = v$);
4:     for $j = 1 \cdots T$ do
5:         $l_{i_j} = l_{i_j} - \prod_{l=1}^{j} w(i_l, i_{l-1})(1 + \sigma_{T-j}^{V \backslash S \backslash \{i_1, \cdots, i_j\}}(v))$;
6:     end for
7: end for
8: output: IL.
```

In Algorithm 3, the incoming tree is node repeated, including all the simple path of length T ending at v. $\prod_{l=1}^{j} w(i_l, i_{l-1})$ denotes the EIS from i_j to i_0 via path $(i_j, i_{j-1}, \cdots, i_0)$, where $i_0 = v$, and $\sigma_{T-j}^{V \backslash S \backslash \{i_1, \cdots, i_j\}}(v)$ denotes the EIS of v in the induce graph of $V \backslash S \backslash \{i_1, \cdots, i_j\}$. Thus, $\prod_{l=1}^{j} w(i_l, i_{l-1})(1 + \sigma_{T-j}^{V \backslash S \backslash \{i_1, \cdots, i_j\}}(v))$ denotes the entire influence diffused from i_j through path $(i_j, i_{j-1}, \cdots, i_1, \pi)$, where π is a path of length no more than $T - j$ starting from v and does not contain any node in $\{i_1, i_2, \cdots, i_j\}$. It is clear that after steps 2 to 7, $\forall u \in (V \backslash S)$, the influence diffused from u through v is removed from the corresponding element in IL. Consider now the running time. Algorithm 3 generates at most $O(\Delta^j)$ nodes in depth j $(1 \leq j \leq T)$. For each node i_j in depth j, $\sigma_{T-j}^{V \backslash S \backslash \{i_1, \cdots, i_j\}}(v)$ can be computed by building an outgoing tree of depth $T - j$ rooted at v, which can be done by DFS in $O(\Delta^{T-j})$ time. Therefore, Algorithm 3 runs in $O(\Delta^T)$ time, considering T as a constant. Compared with running EISE for all the nodes, it is much faster when T and Δ are relatively small.

In addition to IL, we construct another list, namely, probability list *PL*, to store the nodes' active probabilities at time T. When $S = \emptyset$, obviously $\text{PL} = \{p_1 = 0, \cdots, p_n = 0\}$. Similarly, after adding a node v_i into S, the active probabilities of nodes in v_i's local area need to be updated. The algorithm of updating PL is given in Algorithm 4.

Algorithm 4 Update*PL*

```
0: input: $G = (V, E, w)$, $v$, $S$, and PL.
1: construct an outgoing tree of depth $T$ rooted at $v$ in the induced graph of $G \backslash S$
   (without loss of generality, assume the simple paths are $\pi_1, \pi_2, \cdots, \pi_m$);
2: for $i = 1 \cdots m$ do
3:     let $i_0, i_1, \cdots, i_T$ be the nodes visited by $\pi_i$ sequentially and $p_{i_0}, p_{i_1}, \cdots, p_{i_T}$ be the
       corresponding elements in PL;
4:     for $j = 1 \cdots T$ do
5:         $p_{i_j} = p_{i_j} + (1 - p_v) \prod_{l=0}^{j-1} w(i_l, i_{l+1})$;
6:     end for
7: end for
8: output: PL.
```

Algorithm 4 searches the simple paths of length T starting from v and updates the active probability of a node i_j according to step 5, in which $\prod_{l=0}^{j-1} w(i_l, i_{l+1})$ is the influence spread from v to i_j through path $(i_0, \cdots, i_j)$, and $1 - p_v$ is the increment of v' active probability when it is added into S. In the outgoing tree, there are $O(\Delta^T)$ nodes; thus, PL can be updated in $O(\Delta^T)$ time.

Assume v_i is a newly added node; then, the marginal gain is $l_i(1 - p_i)$. Since both Algorithms 3 and 4 run in $O(\Delta^T)$ time, we can find the node with the maximum marginal gain in $O(\Delta^T + n)$ time. Next, we present an algorithm, which consists of two steps, for influence maximization based on a time parameter T (IMT). Given a weighted directed graph $G(V, E, w)$, the first step is to compute the EIS of each node $v \in V$. Such computation is based on the assumption that the EIS is negligible after T hops. The second step contains two parts, the first part is to choose a node with the maximum marginal gain and the second part is to update the two lists: IL and PL. Let v be the last added node; the updating is limited to the local area of v (T hops from v).

The running time of Algorithm 5 highly depends on T and the maximum degree Δ. In [15], when estimating the EIS of a node by searching simple paths, a parameter η is used to prune a path once its influence spread is less than η. To further reduce the running time, when building the incoming and outgoing trees (step 6), we prune the paths in the same way. It is worthy to mention that in [15], the EISE of a node v misses all the outgoing simple paths of v whose product of weights is less than η. When building the incoming (respectively outgoing) tree rooted at v, our algorithm also neglects a number of paths; however, the losses are now evenly distributed to all the nodes in v's local area. Thus, the impact is less significant.

Algorithm 5 IMT

```
0: input: a weighted directed graph $G = (V, E, w)$ and two integers $T$ and $k$.
1: let $S = \emptyset$;
2: let IL be the list resulted by Algorithm 1 and Algorithm 2 and let PL = 0;
3: while $|S| < k$ do
4:     let $v_i$ be the node in $V \backslash S$ that has the maximum $l_i \cdot (1 - p_i)$;
5:     add $v$ into $S$;
6:     update IL and PL by Algorithm 3 and Algorithm 4;
7: end while
8: output: $S$.
```

Results and discussion

We perform three experiments to evaluate the proposed algorithms. The performance metrics are average influence spread (AIS) and program running time (PRT). Since our algorithm is based on a parameter T, we will first analyze how it impacts the time performance and the quality of seed selection. In the second experiment, we will compare IMT (Algorithm 5) with some well-known IM algorithms in terms of AIS. In the last experiment, we will investigate the accuracy of our EISE (Algorithms 1 and 2) and the accuracy of SIMPATH [15]. The data sets used in this paper are introduced in detail in the 'Simulation environments' section, and the algorithms are described in the 'Algorithms' section.

Simulation environments

The experiments are conducted on four real-world networks: 'Hep', 'Phy', 'Amazon', and 'Flixster', which have been widely used for evaluating IM algorithms under different models [5,9-11,15]. The dataset statistics are summarized in Table 1. Briefly, 'Hep' and 'Phy' are academic author networks extracted from http://www.arXiv.org, where nodes denote authors and edges denote collaborations. 'Amazon' is a product network, where nodes denote products and edge (u, v) denote product v which is often purchased with product u. 'Flixster' is a social network allowing users to rate movies, in which nodes denote users and edges denote friendships.

In all types of social networks, let $\deg_{\text{in}}(v) = |N_{\text{in}}(V)|$ be the in-degree of node v; we use a classic method proposed in [5] to add the weights to edges, i.e., $w(u, v) = c(u, v)/\deg_{\text{in}}(v)$, where $c(u, v)$ is the number of edges from u to v.

Algorithms

For the comparison purposes, we evaluate some well-known algorithms designed for IM under the LT model and some model independent heuristics for IM as follows:

- MC: The greedy algorithm with MC simulation and CELF optimization. Each time, we simulate 10K runs to get the EIS of a seed set.
- LDAG: The LDAG algorithm proposed in [11]. As recommended by the authors, the pruning threshold $\eta = \frac{1}{320}$.
- SP: The SIMPATH algorithm proposed in [15]. As recommended by the authors, the pruning threshold $\eta = \frac{1}{1{,}000}$.
- MAXDEG: A heuristic algorithm [5] based on the notion of 'degree centrality', considers higher-degree nodes are more influential.
- PR: The PAGE-RANK algorithm proposed for ranking the importance of pages in web graphs. We can compute the PR value for each node by the power method with a damping value between 0 and 1. In the experiments, it is set to 0.15, and the algorithm stops when two consecutive iterations differ for at most 10^{-4}.
- RANDOM: The RANDOM algorithm chooses the nodes uniformly at random. It was proposed in [5] as a baseline method for comparison purposes.

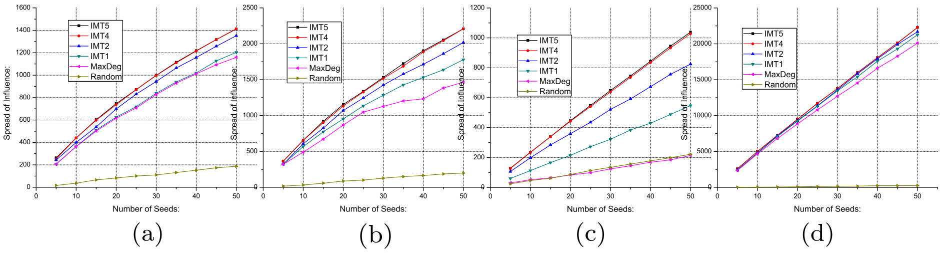
We run 10K MC simulations to approximate the AIS of seed set S resulted by the above algorithms. All the experiments are run on a PC with a 2.6-Ghz processer and 6-GB memory.

Experimental results

To understand how effectively the hop constraint T can help us to balance the algorithm efficiency and quality of seed selection, we run IMT on the four data sets, with T varying

Table 1 Statistics of datasets

Dataset	Hep	Phy	Amazon	Flixster
Number of nodes	12K	37K	257K	720K
Number of edges	60K	348K	1.2 million	10 million
Maximum out-degree	64	178	5	1,010
Maximum in-degree	62	178	420	319



(a) (b) (c) (d)

Figure 5 Simulation results of IMT when *T* varies in the range of [1,5] (spread of influence). (a) Hep, **(b)** Phy, **(c)** Amazon, and **(d)** Flixster.

in the range of $[1, 5]$. The simulation results are shown in Figure 5 and Table 2, in which MaxDeg and Random are considered as baselines. When $T \leq 4$, the EIS is estimated by Algorithm 1; and when $T = 5$, it is estimated by Algorithm 2 with parameter $r = 1,000$. Figure 5 shows the AIS of seed sets resulted by IMT, MaxDeg, and Random. First, the AIS of IMT in all the datasets is non-decreasing as T increases. This agrees with our intuition in that increasing the number of hops brings more accurate EISE. Second, the increments of AIS are tiny when increasing T from 4 to 5, which implies that the seed quality of IMT$_{T=4}$ is as good as that of IMT5. From Figure 5, we also can get that the performance of IMT$_{T=2}$ is much better than that of IMT$_{T=1}$ for the first three data sets, and it is slightly worse than IMT$_{T=4}$. In the 'Flixster' data set, all the algorithms perform similarly, except Random, which is always the worst one in all the experiments.

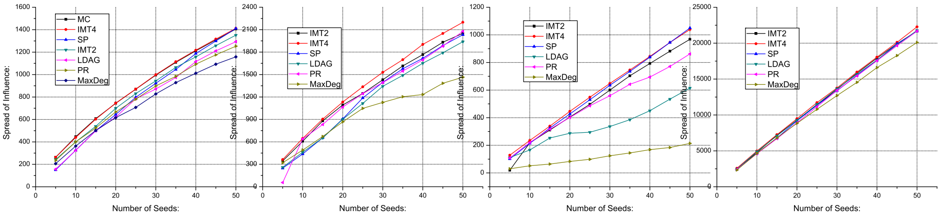
Consider now the running time performance. Table 2 shows the PRT of IMT, in which the file reading and writing time are not counted. When $T \leq 4$, on the first three data sets, IMT is extremely fast, since the maximum out-degree in those data sets is not large. For instance, IMT$_{T=4}$ only requires less than 1 s to finish in 'Hep'. In 'Flixster', IMT is fast when $T \leq 2$, and it is relatively slow when $T \geq 4$. When $T = 5$, the PRT of IMT increases in certain degree for all the data sets. It is reasonable since in such a case, Algorithm 1 does not work, and Algorithm 2 is applied.

According to the first experiment, one notes that, in general, IMT$_{T=4}$ is an efficient algorithm for seed selection. When the running time is of first priority or the data set is extremely large, IMT$_{T=2}$ is a good replacement.

In the second experiment, we compare IMT$_{T=2}$ and IMT$_{T=4}$ with the algorithms introduced in the 'Algorithms' section. The results are exhibited in Figure 6. Since MC is not scalable, its results are omitted for the last three data sets. As shown in Figure 6a, IMT$_{T=4}$ and MC perform similarly in 'Hep'. SP is about 2% lower than IMT$_{T=4}$ and MC in spread achieved when the number of seeds is 35, and its performance matches IMT$_{T=4}$ and MC when the number of seeds is greater or equal to 40. In the other three data sets, IMT$_{T=4}$

Table 2 Running time performance (seconds)

Dataset	Hep	Phy	Amazon	Flixster
IMT$_{T=1}$	0.14	0.28	0.37	1.32
IMT$_{T=2}$	0.26	0.41	0.53	2.57
IMT$_{T=3}$	0.46	0.92	1.01	30.54
IMT$_{T=4}$	0.73	2.44	2.41	126.95
IMT$_{T=5}$	5.71	11.18	81.83	363.42



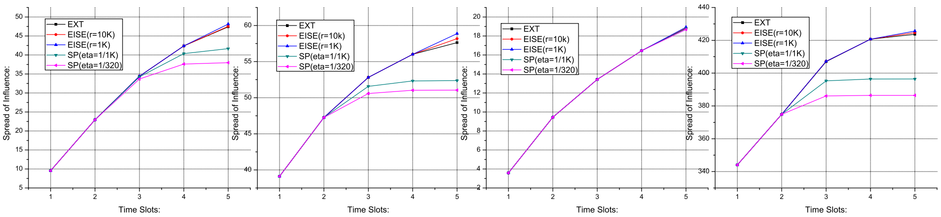
(a) (b) (c) (d)

Figure 6 Simulation results of multiple methods on four datasets (spread of influence). (a) Hep, **(b)** Phy, **(c)** Amazon, and **(d)** Flixster.

is able to produce seed sets of the highest quality, and $\text{IMT}_{T=2}$ is also compatible with other algorithms in terms of AIS. In general, $\text{IMT}_{T=4}$ is the best one. In 'Phy', $\text{IMT}_{T=4}$ outperforms SP by about 0% to 10%, and in 'Amazon' and 'Flixster', they perform similarly. $\text{IMT}_{T=2}$ outperforms PR and LDAG in 'Hep' and 'Amazon', and they perform similarly in 'Phy'. In 'Flixster', all the methods perform well. More than 20K nodes can be activated by the seed set resulted by any algorithm in 'Flixster'. It is probably because there are a lot of high-degree nodes in 'Flixster' (as shown in Table 1, the maximum degree node in 'Flixster' has 1,010 outgoing neighbors).

Although MC is able to produce high-quality seed sets, it is not scabble. In terms of PRT, $\text{IMT}_{T=2}$ is orders of magnitude faster than MC, and $\text{IMT}_{T=4}$ is also much faster than MC. According to the experiments, MC takes 8,532.6 s to finish in 'Hep'. As shown in Table 2, the running time of $\text{IMT}_{T=2}$ and $\text{IMT}_{T=4}$ is only 0.26 and 0.73 s, respectively. Therefore, IMT is much more scalable than MC. In sum, IMT is better than other algorithms in terms of AIS except MC, and it is more suitable than MC for finding seed set in large social networks.

Finally, we would also like to evaluate the accuracy of our EISE algorithms. To do this, we compute the EIS for the most influential node in each data set by our EISE algorithms and by the SP algorithm, respectively. The results are compared with the exact solutions. Figure 7 shows the comparisons, in which 'Ext' denotes the exact EIS_T which is computed by enumerating all the simple paths of length no more than T. Our results exactly match the exact solutions when $T \leq 4$, which validates our conclusion in the 'A deterministic algorithm' section (EISE_4 is exact). For the case that $T = 5$, when $r = 1,000$, the errors of EISE are about 1%, 2%, 0.1%, and 1% in the four data sets, where r denotes the number of uniform random samples. When $r = 10,000$, the error is much lower. Compared with the



(a) (b) (c) (d)

Figure 7 Accuracy of EISE when *T* Varying in the range of [2,5]. (a) Hep, **(b)** Phy, **(c)** Amazon, and **(d)** Flixster.

SP method with a pruning threshold η, EISE is much more accurate in computing the EIS in data sets: 'Hep', 'Phy', and 'Flixster'. In 'Amazon', the results of both EISE and SP match the exact solution. Note that in the second experiment, $\text{IMT}_{T=4}$ outperforms SP in 'Hep' and 'phy', and they perform similarly in 'Amazon'. Thus, we can say that an accurate EISE algorithm is indeed important for solving the IM problem.

Conclusion

IM is a big topic in social network analysis. In this paper, we investigate efficient influence spread estimation for IM under the LT model. We analyze the problem both theoretically and practically. By adding a hop constraint T, we show that the influence estimation problem can be solved efficiently when T is small, and it can be approximated well by uniform random sampling. Based on the two points, we develop a new algorithm called IMT for the LT model. The efficiency of IMT is demonstrated through simulations on four real-world social networks.

In future research, we plan to extend our work to other influence propagation models such as the IC model. Furthermore, we will study constraints under which the optimal solution for IM can be obtained.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

ZL, LF, and KY formulated the problem and did the algorithm design and implementation. WW and BT contributed to the theoretical part of algorithm design and organized this research. All authors read and approved the final manuscript.

Acknowledgements

This research work was supported in part by the US National Science Foundation (NSF) under grants CNS 1016320 and CCF 0829993.

Received: 14 April 2014 Accepted: 22 May 2014
Published online: 15 October 2014

References

1. Domingos, P, Richardson, M: Mining the network value of customers. In: 2001 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 57–66 San Francisco, CA, USA, (August 26-29, 2001)
2. Goldenberg, J, Libai, B, Muller, E: Using complex systems analysis to advance marketing theory development. Acad. Market. Sci. Rev. **9**(3), 1-18 (2001)
3. Goldenberg, J, Libai, B, Muller, E: Talk of the network: a complex systems look at the underlying process of word-of-mouth. Marketing Lett. **12**(3), 211–223 (2001)
4. Richardson, M, Domingos, P: Mining knowledge-sharing sites for viral marketing. In: the 2002 International Conference on Knowledge Discovery and Data Mining, pp. 61–70 Edmonton, AB, Canada, (July 23-25, 2002)
5. Kempe, D, Kleinberg, J, Tardos, É: Maximizing the spread of influence through a social network. In: The 2003 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 137–146 Washington, DC, USA, (August 24-27, 2003)
6. Ma, H, Yang, H, Lyu, MR, King, I: Mining social networks using heat diffusion processes for marketing candidates selection. In: The 2008 ACM Conference on Information and Knowledge Management, pp. 233–242 Napa Valley, CA, USA, (October 26-30, 2008)
7. Granovetter, M: Threshold models of collective behavior. Am. J. Sociol. **83**(6), 1420–1443 (1978)
8. Schelling, T: Micromotives and Macrobehavior. W.W. Norton, New York, USA, (1978)
9. Chen, W, Wang, Y, Yang, S: Efficient influence maximization in social networks. In: The 2009 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 199–208 Paris, France, (June 28 - July 01, 2009)
10. Chen, W, Wang, C, Wang, Y: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: The 2010 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1029–1038 Washington DC, DC, USA, (July 25-28, 2010)
11. Chen, W, Yuan, Y, Zhang, L: Scalable influence maximization in social networks under the linear threshold model. In: The 2010 International Conference on Data Mining, pp. 88–97 Sydney, Australia, (December 14-17, 2010)
12. Narayanam, R, Narahari, Y: A Shapley value based approach to discover influential nodes in social networks. IEEE Trans. Automation Sci. Eng. **8**(1), 130–147 (2011)
13. Goyal, A, Lu, W, Lakshmanan, LVS: CELF++: optimizing the greedy algorithm for influence maximization in social networks. In: The 2011 International World Wide Web Conference, pp. 47–48 Hyderabad, India, (March 28 - April 01, 2011)

14. Jiang, Q, Song, G, Cong, G, Wang, Y, Si, W, Xie, K: Simulated annealing based in influence maximization in social networks. In: The 2011 AAAI Conference on Artificial Intelligence. San Francisco, CA, USA, (August 7-11, 2011)
15. Goyal, A, Lu, W, Lakshmanan, LVS: SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In: The 2011 IEEE International Conference on Data Mining, pp. 211–220 Vancouver, Canada, (December 11-14, 2011)
16. Leskovec, J, Krause, A, Guestrin, C, Faloutsos, C, VanBriesen, J, Glance, NS: Cost-effective outbreak detection in networks. In: The 2007 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 420–429 San Jose, CA, USA, (August 12-15, 2007)
17. Nemhauser, G, Wolsey, L, Fisher, M: An analysis of the approximations for maximizing submodular set functions. Math. Program. **14**(1978), 265–294 (1978)
18. Kimura, M, Saito, K, Nakano, R: Extracting influential nodes for information diffusion on social network. In: The 2007 AAAI Conference on Artificial Intelligence, pp. 1371–1376 Vancouver, British Columbia, (July 22-26, 2007)
19. Wang, Y, Cong, G, Song, G, Xie, K: Community-based greedy algorithm for mining top-*k* influential nodes in mobile social networks. In: The 2010 ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1039–1048. Washington DC, DC, USA, (July 25-28, 2010)
20. Hoeffding, W: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963)

doi:10.1186/s40649-014-0002-3
Cite this article as: Lu *et al.*: **Efficient influence spread estimation for influence maximization under the linear threshold model.** *Computational Social Networks* 2014 **1**:2.