

Incremental Influence Maximization for Dynamic Social Networks

Yake Wang¹, Jinghua Zhu^{1,2(✉)}, and Qian Ming^{1,2}

¹ School of Computer Science and Technology,
Heilongjiang University, Harbin 150001, China
zhujinghua@hlju.edu.cn

² Key Laboratory of Database and Parallel Computing of Heilongjiang Province,
Harbin, China

Abstract. Influence maximization is one fundamental and important problem to identify a set of most influential individuals to develop effective viral marketing strategies in social network. Most existing studies mainly focus on designing efficient algorithms or heuristics to find Top-K influential individuals for static network. However, when the network is evolving over time, the static algorithms have to be re-executed which will incur tremendous execution time. In this paper, an incremental algorithm DIM is proposed which can efficiently identify the Top-K influential individuals in dynamic social network based on the previous information instead of calculating from scratch. DIM is designed for Linear Threshold Model and it consists of two phases: initial seeding and seeds updating. In order to further reduce the running time, two pruning strategies are designed for the seeds updating phase. We carried out extensive experiments on real dynamic social network and the experimental results demonstrate that our algorithms could achieve good performance in terms of influence spread and significantly outperform those traditional static algorithms with respect to running time.

Keywords: Influence maximization · Dynamic social network · Linear threshold model · Pruning strategy

1 Introduction

In recent years, large social networks have sprung up not only as an fundamental medium for people to exchange information, make friends, but also as an important business platform allowing businessmen to display and sell merchandise. In order to reach the largest scope of products advertisement, businessmen usually choose a small part of influential people in social networks and provide them with free products to make them recommend products to their friends. For example, some business men provide free commodities for some stars, because they usually have a lot of fans in the social network, their recommendations will enhance the product reputation among fans who may be interested in these products and buy them.

Although many existing researches have proposed a number of efficient algorithms for influence maximization, but most of them are based on static social networks. As a

matter of fact, real-world social networks keep evolving over time, new accounts are created and some users will establish or lose contacts, the quantity of users and relationships are constantly changing. What’s more, no measure can accurately predict the dynamic evolve of social networks because of two reasons: first, the relationships between users are randomly established; second, those relationships may change. Previous static algorithms can’t capture and deal with these topology changes.

Although one could possibly run any of the static influence maximization algorithms to find the new Top-K influential nodes when the social network changes, the running time of the static algorithms on large scale social network will be extremely long and whenever the network topology changes, we need to recalculate the influence spreads for all nodes which will lead to quite high cost.

To address the challenges posed by the rapidly and unpredictable changing topology for dynamic social network, we proposed an efficient incremental influence maximization algorithm especially for dynamic social networks called Dynamic Influence Maximization (DIM) in this paper. DIM algorithm includes two stages: initial seeding (Init_Seeding) and incrementally seed updating (Inc_SeedUpdate). At time $t = 0$, we run Init_Seeding algorithm to get the initial seeds set for the static social network. Init_Seeding first gets simple paths by travelling network graph and then calculates the influence spread for each node, finally it outputs the initial seeds and derives the necessary conditions for the second phase. At the subsequent time t , we run Inc_SeedUpdate algorithm continuously to update the seeds. Inc_SeedUpdate calculates the influence spread change of nodes caused by the network change efficiently and quickly finds the new Top-K influential nodes in the evolution based on previously known information. In order to narrow the search space into nodes only experiencing major spread change, we put forward two pruning strategies: influence value increment pruning strategy and degree pruning strategy. The experimental results show that DIM algorithm can achieve as much as 30 speedup in execution time comparing to the static influence maximization algorithm while maintaining good performance in terms of influence spread.

To summary, the main contributions of this paper are as follows:

- We design an incremental influence maximization algorithm under Linear Threshold Model called DIM for dynamic social network.
- We propose two pruning strategies: influence value increment pruning strategy and degree pruning strategy to narrow the search space and improve the time efficiency of DIM algorithm.
- We conduct extensive experiments on NetHEPT, Facebook, Flixster and Flickr social network and the experimental results show that our algorithms can achieve 30 speedup in execution time while providing matching influence spread compared with the state-of-the-art static algorithms.

The rest of this paper is organized as follows. Section 2 reviews related work, Sect. 3 gives some preliminaries and the problem statement. We introduce the dynamic influence maximization algorithm DIM in Sect. 4. The experimental results as well as the analysis are given in Sect. 5. We make conclusions and outline future works in Sect. 6.

2 Related Work

Influence maximization on static networks has attracted a lot of attentions. Viral marketing, first introduced to by Richardson and Domingos [1] is a significant marketing strategy that promotes commodity by giving free or discount products to a small subset of influential users. Kempe [2] proved that the influence maximization problem is NP-Hard and proposed hill-climbing greedy algorithm with a provable approximation ratio $(1-1/e-\epsilon)$. Leskovec [3] proposed CELF algorithm which is 700 times faster than the greedy algorithm. Chen [4] proposed heuristic algorithm MIA to enhance the scalability of the algorithm using maximum influence path of each pair of nodes. Kim et al. [5] proposed a scalable and parallelizable influence approximation algorithm based on independent paths. Yang et al. [6] extended the influence maximization problem and proposed a coordinate descent method to solve the transmission cost problem. All the above algorithms are designed for static networks, without considering the dynamic changes of the network.

There are a few influence maximization algorithms designed for dynamic networks. Chen [7] proposed a dynamic social network model which keeps involving during the influence propagation. Zhuang et al. [8] argued that the changes of the network can be obtained by traversing some probing nodes whose topological changes can approximately reflect the evolution of the whole network. You et al. [9] found that under certain incentives, constructing new relationships in social network are helpful for the influence diffusion process. Tong et al. [10] modeled the influence diffusion in social network as the probability event, both the activation of node and the probability on edges obey a certain distribution, influence spread is the expected number of the probability event. Liu et al. [11] proposed an incremental approach to identify the top-K influential individuals based on maximum influence path MIA.

All the above static or dynamic influence maximization algorithms are under the Independent Cascade Model. Goyal et al. [12] used the simple path between neighbor nodes to estimate the influence propagation spread. Lu et al. [13] presented an approximation algorithm to estimate the influence spread, an exact algorithm to compute the influence spread of node within four step.

3 Preliminaries and Problem Statement

3.1 Linear Thread Model

In this paper, a social network is represented as a weighted directed graph $G = (V, E, W)$, here V is node set, each node represents an individual in social network. $E \subseteq V \times V$ is edge set, each edge represents relationships between individuals. For example, a directed edge (v_i, v_j) will be established from node v_i to v_j if v_i is followed by v_j which represents that v_j may be influenced by v_i . $W: E \rightarrow [0, 1]$ represents the influence probability on edges, each edge $(e_i, e_j) \in E$ is associated with an influence probability $w(e_i, e_j)$, $\sum_{u \in V} w(u, v) \leq 1$. Nodes in network have two states: active or inactive. Once a node is activated, it will try to activate its inactive neighbors, if the activation succeeds, its neighbors will become active. Influence diffusion model is the operational model for

the spread of an idea or innovation through a social network, which is the basis of influence maximization research. Currently, there are two diffusion models IC (Independent Cascade Model) and LT (Linear Threshold Model). In this paper, we study the influence maximization problem under LT model.

LT model is based on the use of node-specific threshold. Because the active neighbors co-decide whether other nodes will become active, the activation of nodes are independent and satisfy $\sum_{u \in pre(v)} w(u, v) \leq 1$, here $pre(v)$ represent precursor node set of node v , $w(u, v)$ is the probability that u successfully active v .

Each node v chooses a threshold θ_v , $\theta_v \in [0, 1]$, that intuitively represents the different latent tendency of nodes to adopt the innovation when their neighbors do. If θ_v is big then activation of node v is difficulty, on the contrary, activation is easy. At step $t = 0$, only nodes in $S \subseteq V$ is active. If x become active at step $t-1$, x may activate its inactive out-neighbor v . Vertex v is activated at step t only if the weighted number of its activated in-neighbor reaches its threshold, i.e. $\sum_x w(x, v) \geq \theta_v$, here x represents the active nodes in $pre(v)$. This process stops when no more node can be activated. We use $\sigma(S)$ to denote the influence spread of the initial seed set S , $\sigma(S)$ can be approximated by the expected number of active nodes in S .

Kempe et al. [2] proposed live-edge model: given an influence graph $G = (V, E, W)$, for every vertex v , select at most one of its incoming edges at random, such that edge (u, v) is selected with probability $w(u, v)$, and no edge is selected with probability $1 - \sum_u w(u, v)$. The selected edges is called live edge and all other edges are called blocked edge. G_L denotes the spanning subgraph which includes all vertices in V and all live edges selected. If vertex u can reach vertex v in G_L , there exists live path from u to v which consists of all live edges. Kempe et al. [3] proved that, given a seed set S , Linear Threshold Model and live-edge model can achieve the same influence spread.

$$\sigma(S) = \sum_{G_L} pro[G_L] \cdot \sigma_{G_L}(S) \quad (1)$$

Here $pro[G_L]$ represents the probability of live-edge graph G_L appeared, $\sigma_{G_L}(S)$ denotes the expected number of active nodes starting from S in G_L .

3.2 Influence Maximization

Given an influence graph $G = (V, E, W)$ and integer k , influence maximization problem is to find a set of top- k influential nodes in social network so that their aggregated influence is maximized as shown in this formula:

$$S^* = \arg \max_{S \subseteq V, |S|=k} \sigma(S) \quad (2)$$

Kempe et al. [2] proved the influence maximization problem is NP-Hard under Linear Threshold Model and proposed a greedy algorithm to solve it. The marginal influence (MI) of any node $v \in V$ given seed set S is defined as $MI(v|S) = \sigma(S \cup \{v\}) - \sigma(S)$. They used Monte-Carlo simulation to estimate the influence spread. The monotonicity and submodularity of $\sigma(S)$ guarantee that the greedy algorithm has approximation ratio $(1 - \frac{1}{e} - \varepsilon)$, that is, it returns a seed set S^g such that $\sigma(S^g) \geq (1 - \frac{1}{e} - \varepsilon)\sigma(S^*)$, for any small $\varepsilon > 0$, where ε accommodates the inaccuracy in Monte-Carlo estimation.

3.3 Influence Maximization for Dynamic Network

Real-world social networks are not static and keep evolving over time. Rapid changes in social networks have brought many new challenges to influence maximization problem, one of the most interesting and urgent problem is that how to quickly identify the most influential users in dynamic social network. So this paper focus on studying the influence maximization in dynamic network.

Dynamic Influence Maximization (DIM): We define an evolving network $G^S = \{G^1, G^2, G^3, \dots, G^t\}$ as a sequence of network snapshots evolving over time, where $G^t = (V^t, E^t, W^t)$ is an influence graph snapshot at step t , $\Delta G^t = (\Delta V^t, \Delta E^t, \Delta W^t)$ represents the topological change of network graph G^t , obviously, $G^{t+1} = G^t \cup \Delta G^t$. Given an influence graph G^t at step t , the topological change ΔG^t of network graph G^t , the top-K influential nodes S^t in G^t , then we identify the influential nodes $S^{t+1} \subset V^{t+1}$ of size K in G^{t+1} at time $t + 1$.

When network changes, using traditional static influence maximization algorithm to find the seed set will lead to large computational overhead. This paper proposes an incremental algorithm DIM, which identifies the most influential nodes set for dynamic networks by incremental method.

4 Dynamic Influence Maximization Algorithm DIM

The DIM algorithm proposed in this paper is divided into two stages. The first stage is to obtain the initial seed set in static network and the second stage is to incrementally update the initial seed set after the network evolved. In order to further reduce the computational overhead, two optimization strategies are proposed in Sect. 4.3.

4.1 Init_Seeding Algorithm

Social networks keep evolving over time. But if we observe the changes of social network from discrete time step perspectives, in each time t , the social network topology is a static graph. At time $t = 0$, social network is an unchanged initial graph, according to formula (2), the influence spread of seed set S in influence graph G is $\sigma(S) = \sum_{G_L} pro[G_L] \cdot \sigma_{G_L}(S)$, here:

$$\sigma_{G_L}(S) = \sum_{v \in V} I_{G_L}(S, v) \quad (3)$$

$I_{G_L}(S, v)$ represents whether there exists live path from S to v in G_L . If live path is existed then $I_{G_L}(S, v) = 1$, otherwise 0. Therefore,

$$\sigma(S) = \sum_{v \in V} \sum_{G_L} pro[G_L] \cdot I_{G_L}(S, v) = \sum_{v \in V} \sigma(S, v) \quad (4)$$

$\sigma(S, v)$ is the activation probability of vertex v after S is selected as seed set, that is, the influence of set S on node v . Nodes in S may active node v through the path in live-edge model. Let $P = \langle v_u, v_1, \dots, v_m \rangle$ denote a path, $(v_i, v_j) \in P$ indicates edge (v_i, v_j)

belongs to path P . The simple path is a path whose nodes are all different. The probability of a path P being live path is defined as follows:

$$pro[P] = \prod_{(v_i, v_j) \in P} w(v_i, v_j) \quad (5)$$

The influence of vertex u on v is:

$$\sigma(u, v) = \sum_{P \in Path_{u,v}} pro[P] \quad (6)$$

Here $Path_{u,v}$ is the path set from node u to node v . Therefore, the influence spread of node u can be represented as follows:

$$\sigma(u) = \sum_{v \in V} \sigma(u, v) \quad (7)$$

As an example, we consider the influence graph shown in Fig. 1. Assume that the probability of each edge is 0.2, then the influence of node 1 on node 4 is: $\sigma(v_1, v_4) = pro[P = \langle 1, 2, 3, 4 \rangle] + pro[P = \langle 1, 2, 3, 5, 4 \rangle] = 0.008 + 0.0016 = 0.0096$, and the influence spread of node 1 is:

$$\begin{aligned} \sigma(v_1) &= \sigma(v_1, v_1) + \sigma(v_1, v_2) + \sigma(v_1, v_3) + \sigma(v_1, v_4) + \sigma(v_1, v_5) + \sigma(v_1, v_6) + \sigma(v_1, v_7) \\ &= 1 + 0.2 + 0.04 + 0.0096 + 0.008 + 0.0016 + 0.0016 = 1.2608. \end{aligned}$$

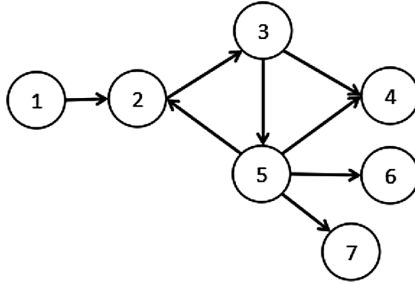


Fig. 1. Example of influence graph

The influence spread of set S is the sum of the spread of each node u in S .

$$\sigma(S) = \sum_{u \in S} \sigma^{V-S+u}(u) \quad (8)$$

The marginal influence $MI(v)$ of node v under seed set S , $MI(v|S) = \sigma(S \cup \{v\}) - \sigma(S)$. To compute $\sigma(S \cup \{v\})$, we need to compute $\sigma^{V-S-v+x}(S \cup \{v\})$ for each $x \in S \cup \{v\}$ on subgraph induced by $V - S - v + x$ according formula (4-6). Each node in S except node x , when adding new node to S , needs recalculation using formula

(4–6) even the basic subgraph has slight difference. Goyal et al. [18] proposed that the expected number of influence node will not be affected too much for slightly changed subgraph. The calculation formula of $\sigma(S \cup \{v\})$ is derived as follows:

$$\begin{aligned}\sigma(S \cup \{v\}) &= \sum_{x \in S \cup \{v\}} \sigma^{V-S-v+x}(x) = \sigma^{V-S}(v) + \sum_{x \in S} \sigma^{V-S-v+x}(x) \\ &= \sigma^{V-S}(v) + \sum_{x \in S} \sigma^{V-S-v+x}(x) = \sigma^{V-S}(v) + \sigma^{V-v}(S)\end{aligned}\quad (9)$$

For example, as shown in Fig. 1, if seed set $S = \{v_5\}$, then $\sigma(S) = \sigma(v_5) = 1.848$. After node v_3 is incorporated into S , $\sigma(S \cup \{v_3\}) = \sigma^{V-v_5}(v_3) + \sigma^{V-v_5}(v_5) = 1.2 + 1.8 = 3.0$.

At time $t = 0$, the social network is an unchanged initial graph, we call algorithm GetAllPath to determine the simple path set and compute the influence spread for each node. The problem of enumerating all simple paths in graph is #P-Hard [5]. In social network, the probability of a path being live decreases rapidly as the length of the path increases. Thus, the majority of the influence can be captured by exploring the paths within a small neighborhood, where the size of the neighborhood can be controlled by the error tolerated. Parameter η represents a tradeoff between efficiency and accuracy. Given a probability threshold η , we can filter out those paths whose probabilities are lower than η . In this way, we can efficiently control the size of path set and reduce the computation cost.

GetAllPath algorithm is based on classical backtrack idea to enumerate all simple paths in graph. It starts from node u and traverses u 's out-neighbors in depth-first fashion. If the probability of current path $P = \langle u, \dots, x, v \rangle$ belows η , the algorithm will backtrack to node x , if node x has other out-neighbors, it will continue to traverse follow those out-neighbors, otherwise it will backtrack. In the process, the influence of node u can be estimated by accumulating those probabilities of paths. If it backtracks to starting node u , then the algorithm stops.

Since the probability of path decreases rapidly with increasing length, intuitively, for each change in the graph, the nodes affected by the topology change must be in a small neighborhood. In order to accurately and quickly determine which nodes are affected by topology change, we compute $IN[v]$ for each $v \in V$ which represents the predecessors of node v . Nodes in $IN[v]$ may influence v within the control threshold η . At the beginning, $IN[v]$ is null for each node, the idea of UpdateIN(v) algorithm that update IN set of node is described as follows: for each node $v \in V$, when the path set of v changed, each path in $Path[v]$ will be traversed. If the path contains node u , which means that v can reach u by the path, then add node v to $IN[u]$ set. If $IN[u]$ already contains node v , it is no need to add v . In such a way, we get all the nodes arriving at node u through the path. When the network is evolving, we can quickly confirm those influenced nodes that need to re-estimate influence spread using IN , we don't have to traverse the entire network and save a lot of running time.

Algorithm Init_Seeding calculates the influence spread of each node at the beginning by calling algorithm GetAllPath, and updates IN set according $Path$ set. In each iteration, according to the greedy idea, the node with the largest marginal influence are added to the seed set until a given size is reached.

The pseudo code of Init_Seeding is described as follows:

Init_Seeding (G, η, k)
Input: graph G , path length threshold η , size k
Output: seed set S

- 1) for each $u \in V$ do
- 2) calculate $\sigma(u)$ by calling GetAllPath algorithm;
- 3) Update $IN(u)$;
- 4) add u to CELF queue;
- 5) $S \leftarrow \emptyset$; $spd \leftarrow 0$;
- 6) While $|S| < k$
- 7) $u \leftarrow$ top node in CELF queue;
- 8) for $s \in S$ //compute $\sigma^{V-u}(S)$
- 9) for each $P \in Path[s]$ and $u \in P$
- 10) $\sigma^{V-u}(s) \leftarrow \sigma^{V-u}(s) - \sigma^{V-u}(P)$;
- 11) $spd \leftarrow \max(spd, \sigma^{V-u}(s))$;
- 12) for each $P \in Path[u]$ and $P \cap S \neq \emptyset$ //compute $\sigma^{V-S}(u)$
- 13) $\sigma^{V-S}(u) \leftarrow \sigma^{V-S}(u) - \sigma^{V-S}(P)$;
- 14) compute $\sigma(S+u)$ by using formula(4-7);
- 15) $MI(u) = \sigma(S+u) - spd$;
- 16) if $MI(u) > \sigma(u)$
- 17) $S \leftarrow S \cup u$; remove u from CELF queue;
- 18) else
- 19) re-insert u in CELF queue;

return S

In line 1–4, we calculate the influence spread of each node in graph G by calling GetAllPath algorithm, update IN set for each node and add them into CELF queue. We initialize seed set S and influence spread spd in line 5 and get the top node in CELF queue in line 7. In line 8–11, as the path set of each node has been obtained, we calculate the influence spread of seed node in the subgraphs induced by $V - u$ equates to making the path contains u starting from the seed node invalid, then we calculate $\sigma^{V-u}(s)$ by subtracting the influence probability of invalid path from the original influence, and compute spd according to the formula (4–6). We compute the influence of node u in subgraphs induced by $V - S$ in line 12–13 and estimate the marginal influence in line 14–15. In line 16–19, if $MI(u)$ of node u is larger than the top node in queue, we directly add this node to seed set, otherwise add it into CELF queue again.

4.2 Inc_SeedUpdate Algorithm

In Social network, new users might join while old users might withdraw, and people will make new friends with each other, thus the topology of network keep evolving. We define an evolving network as $G^S = \{G^1, G^2, G^3, \dots, G^t\}$ from the perspective of discrete time, where $G^{t-1} = (V^{t-1}, E^{t-1}, w^{t-1})$ is the influence graph at step $t - 1$, $\Delta G^{t-1} = (\Delta V^{t-1}, \Delta E^{t-1}, \Delta w^{t-1})$ is the topological change of network graph G^{t-1} , thus $G^t = G^{t-1} \cup \Delta G^{t-1}$. So the influence graph in step t is new relating to the last time $t - 1$ for any $t > 0$. Liu et al. [16] propose six types of topology change as AddEdge,

DelEdge, AddVertex, DelVertex, IncWeight, DecWeight. AddEdge(u, v, w) is to establish a link with weight w from user u to v . DelEdge(u, v, w) is to delete a edge from user u to v whose weight is w . AddVertex and DelVertex correspond to add or delete nodes respectively. IncWeight(u, v, w) is to increase the weight of edge(u, v) to w , and DecWeight(u, v, w) is to decrease the weight to w . For those six changes of the social network, we only explain AddEdge and DelEdge operation in this paper because the other operations are similar.

The pseudo code of AddEdge(u, v, w) is as follows:

AddEdge (u, v, w)
Input: new edge (u, v, w)
Output: the influence spread change $\sigma^\Delta(x), x \in C$

- 1) for each $x \in C$ do
- 2) $\sigma^\Delta(x)=0$;
- 3) for $P_{x,u} \in Path[x]$ do
- 4) if $pro[P_{x,u}] \cdot w(u, v) > \eta$ and $P_{x,v}$ is a simple path
- 5) add $P_{x,v}$ into $temp$;
- 6) if $P_{x,v} \cap S = \emptyset$
- 7) $\sigma^\Delta(x) += pro[P_{x,v}]$;
- 8) if $temp \neq \emptyset$
- 9) for $P_{x,v} \in temp$ do
- 10) for $P_v \in Path[v]$ do
- 11) if $pro[P_{x,v}] \cdot pro[P_v] > \eta, (u, v) \in P_x$
- 12) add P_x into $temp$;
- 13) if $P_x \cap S = \emptyset$
- 14) $\sigma^\Delta(u) += pro[P_x]$;
- 15) copy $P \in temp$ into $Path[x]$;
- 16) UpdateIN(x);

return $\sigma^\Delta(x)$

In AddEdge algorithm, we use parameter w to represent the probability of the new edge (u, v), set C consists of nodes affected by network change, $temp$ is used to temporarily store the new path, $\sigma^\Delta(u)$ is the influence spread change of node u , $P_{x,u}$ represents the path from the node x to node u , and P_v represents the path whose source node is v . We compute influence spread change $\sigma^\Delta(u)$ for each $u \in C$ and update IN set according to the latest $Path$ collection. In line 3–7, we find one path from node x to node u and expand the path by adding new edge (u, v). If the probability of the new path from x to v is greater than threshold η and it is a simple path, then we add this path to $temp$. In line 8–14, if $temp$ is null, that means the probability of those paths are too small or there exists ring among those paths, then the computation will end. Otherwise,

we try to connect the path $P_{x,v}$ with P_v , and perform the same operation. We store those new paths into $Path[x]$ set in line 15, and update $IN[x]$ set based on new $Path[x]$ set of node x in line 16.

Now we analyze the time complexity of AddEdge algorithm. Assume that the maximum number of paths in the path set is P_{\max} , the number of temporary path is up to $O(P_{\max})$ in *temp*, that is, original paths all can be extended. Then the time complexity of circulate in line 3–7 is $O(P_{\max})$, the time complexity of dual-layer cycle in line 8–15 is $O(P_{\max} \cdot P_{\max})$, the time complexity of updating IN in line 16 is $O(P_{\max})$, so the time complexity is $O(|C| \cdot P_{\max}^2)$.

The pseudo code of DelEdge is as follows:

DelEdge (u, v, w)
Input: an edge (u, v)
Output: the influence spread change $\sigma^\Delta(x), x \in C$

- 1) for $x \in C$ do
- 2) $\sigma^\Delta(x) = 0$;
- 3) for $P_x \in Path[x]$ and $(u, v) \in P_x$
- 4) $\sigma^\Delta(x) -= \text{pro}[P_x]$; delete P_x ;
- 5) UpdateIN(x);

return $\sigma^\Delta(x)$

In DelEdge algorithm, we confirm all invalid paths of node x that is influenced by the change, and compute influence spread change $\sigma^\Delta(x)$ of node x and delete all invalid paths. We update IN set based on new $Path$ set of node x in line 5. The time complexity is $O(P_{\max})$ for line 3–5. The time complexity of DelEdge is $O(2 \cdot |C| \cdot P_{\max})$.

The pseudo code of seed set update algorithm Inc_SeedUpdate is as follows:

Inc_SeedUpdate (S^{t-1}, G^{t-1}, G^t, k)
Input: seed set S^{t-1} , graph G^{t-1} and G^t in time $t-1$ and t , size k
Output: seed set S^t in time t

- 1) $S^t \leftarrow \emptyset$;
- 2) While $|S^t| < k$
- 3) for each change c from G^{t-1} to G^t do
- 4) obtain influenced node set C
- 5) for each $u \in C$ do
- 6) compute $\sigma^\Delta(u)$;
- 7) $S^{C^t} \leftarrow S^C \cup u$;
- 8) for each $u \in S^{C^t}$ do
- 9) $MI(u) = \sigma(S^t \cup \{u\}) - \sigma(S^t)$;
- 10) $S^t \leftarrow S^t \cup \text{argmax} MI(u)$;

return S^t

In Inc_SeedUpdate algorithm, S^C is the candidate set, S^t is the seed set at step t . We select seed only from S^C , which can narrow the search space and reduce calculation cost. In line 3–7, according to the six types of topology change, we call the

corresponding algorithms to compute influence spread change of the node, and filter candidate node. In line 8–10, we select the seed node with the largest MI .

4.3 Pruning Strategies

Real social networks like Facebook and Flickr usually have a huge number of users, however the size of seeds set is relatively small for financial reason. Each iteration to compute influence spread of all nodes might cause enormous cost. Although DIM algorithm only re-evaluate the influence spread of nodes affected by network change, those nodes who has small influence spread change still has little chance to be seed nodes. In order to further narrow the search space of influential nodes, we put forward two pruning strategies for DIM algorithm.

Least Influence Increment Pruning Strategy (LIIP): In the i th iteration, the influence spread change of seed set S^{t-1} in graph G^{t-1} is positive, if the influence spread change of node v is greater than any seed node in S^{t-1} set, node v is reserved as the candidate node, otherwise it will be filtered out. In most cases, the most influential nodes in graph G^{t-1} attract large number of nodes and create new connections. Thus the influence spread increase and the influence spread change are positive. If the influence spread and influence spread change of node v are lower than that of any seed node in S^{t-1} at time $t - 1$, then node v must be lower than any seed node in S^{t-1} at time t , that means node v will never become seed node and should be filtered out. Liu et al. [16] find that social network is based on the preferential attachment principle, so the least influence increment pruning strategy can filter out a large number of nodes.

Degree Pruning Strategy (DP): According to the preferential attachment principle, the new-coming edges prefer to attach to nodes with higher degree [16], thus nodes with large degree are more easily influenced by the topology change than nodes with small degree. If the influence spread change of S^{t-1} is negative, it means that the influence spread of S^{t-1} is decreased. In addition to the first optimization measure, the reservation node should also meet one of the following two conditions: First, Degree of nodes is among the top 5% of all nodes in G^t . Second, Degree increase ratio of nodes is among the top 5% of all nodes in G^t . The degree increase ratio of node v is defined as $degree^t(v)/degree^{t-1}(v)$, where $degree^t(v)$ represents the degree of node v in G^t .

We can extend the Inc_SeedUpdate algorithm by adding **LIIP** or **DP** pruning strategy before line 7. We only choose the valid candidate nodes to join S^C . The pruning strategies can further narrow the candidate nodes and improve the efficiency of the algorithm while guaranteeing relatively high accuracy. We call the extended DIM algorithm with pruning strategy Opt-DIM algorithm.

5 Experiments

5.1 Experimental Setup

In this section, we compare our algorithm with the static algorithms in terms of efficiency and effectiveness. We examine two metrics, influence spread and running time.

Influence spread is the final expected number of influenced nodes activated by seed nodes, representing the accuracy of different algorithm; running time is the time to identify the most influential k nodes, reflecting efficiency of algorithm.

We choose four datasets to test the performance of different algorithms, Tables 1 and 2. summarize the statistical information of the four datasets, the growth rate of nodes and edges can intuitively reflect that the real-world social network are rapidly change with time.

Table 1. Table of node information

Datasets	Nodes		
	Initial number	Final number	Growth
NetHEPT	15,634	18,557	18.7%
Facebook	59,736	83,983	40.3%
Flixster	99,825	147,328	48.5%
Flickr	771,738	1,037,995	34.5%

In this paper, we compare our algorithm with two static influence maximization algorithm LDAG and SIMPATH. LDAG algorithm estimate the influence spread of node based on local directed acyclic graph [17]. SIMPATH algorithm finds the Top-K influential nodes based on simple path [18].

Table 2. Table of edge information

Datasets	Edges		
	Initial number	Final number	Growth
NetHEPT	62,836	89,415	42.3%
Facebook	576,653	994,149	72.4%
Flixster	978,265	1,811,249	85.2%
Flickr	4,938,687	7,106,122	43.8%

The active threshold of node v is generated uniformly at random in range $(0, 1)$. We set the weight of every incoming edge of v to be $\frac{1}{d_v}$.

5.2 Experimental Results and Analysis

The first group experiments evaluate the running time to identify 50 most influential nodes of different algorithms. DIM and Opt-DIM call `Init_Seeding` only on the first snapshot, and call `Inc_SeedUpdate` in the following seven network snapshots. As illustrated in Figs. 2 3, 4 and 5, the running time of DIM and Opt-DIM on the first snapshot are longer than on the other snapshots, and much shorter than the other two algorithms on the four datasets. Among the four compared algorithms, LDAG algorithm has the largest time cost. While the running time of DIM algorithm and Opt-DIM

algorithm are relatively stable. Compared with the static algorithm LDAG and SIMPATH, when the dynamic network changes, dynamic algorithm DIM and Opt-DIM are 6.3 times, 13.6 times, 17.4 times and 31 times faster on average than static algorithm on the four datasets.

Although the time costs of DIM and Opt-DIM algorithm are also increased in large scale networks, but the running time still have obvious advantages in large-scale dynamic network. And the perform of Opt-DIM algorithm is better than DIM algorithm on four datasets. On Facebook dataset, Opt-DIM algorithm is 1 time faster than DIM algorithm, while achieving double speedup on Flixster and Flickr datasets on average, that means the pruning strategies are effective.

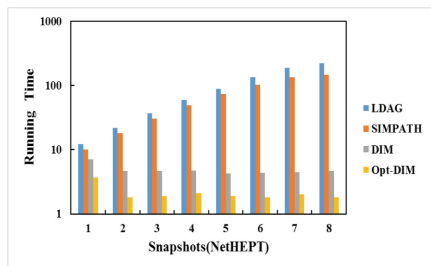


Fig. 2. Running time on NetHEPT dataset

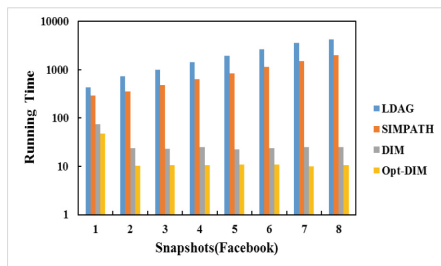


Fig. 3. Running time on Facebook dataset

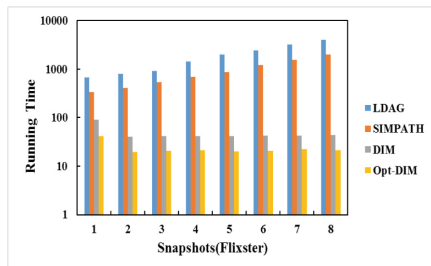


Fig. 4. Running time on Flixster dataset

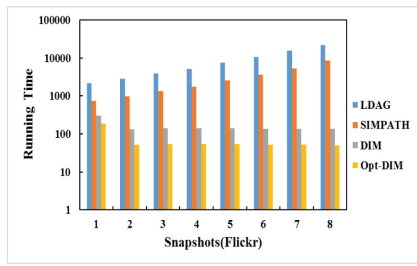


Fig. 5. Running time on Flickr dataset

We use influence spread to measure the accuracy of different algorithms. The influence spread is the number of nodes influenced by the top-50 influential nodes. Figures 6 7, 8 and 9 show the results of the experiments. As shown in Figs. 6 and 7, DIM and Opt-DIM are basically consistent with the static algorithms on NetHEPT and Facebook datasets. However, in Figs. 8 and 9, the influence spread of static algorithm LDAG and SIMPATH have a slight advantage over dynamic algorithms on larger datasets, Flixster and Flickr. This is because LDAG algorithm and SIMPATH algorithm have to estimate the influence spread of all nodes in network in each iteration. This also explains that DIM and Opt-DIM can lift the execution speed ratio at the cost of a little accuracy decrease. The influence spread of SIMPATH algorithm outperforms

that of LDAG algorithm on Flixster and Flickr datasets as shown in Figs. 8 and 9. On the Flixster and Flickr datasets, the advantage of DIM algorithm is more obvious because the pruning strategy based on influence spread change can narrow the search space of influential nodes.

The influence spread of DIM algorithm and Opt-DIM algorithm are smaller than algorithm SIMPATH 4.1%, 5.2% and 5.9% on the Facebook, Flixster and Flickr datasets, respectively. Although the accuracy of the dynamic influence maximization algorithm DIM and Opt-DIM is lower than the static algorithms, according to the operation time showed from Figs. 2, 3, 4 and 5, the dynamic DIM algorithm is more suitable for solving influence maximization problem in large social network.

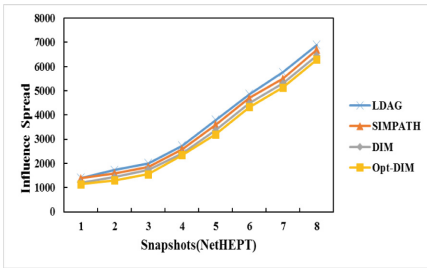


Fig. 6. Influence spread of NetHEPT

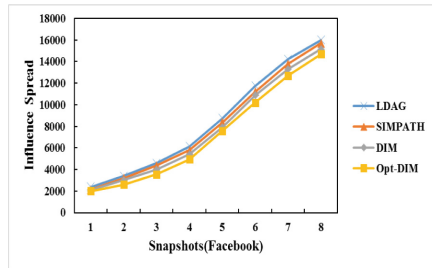


Fig. 7. Influence spread of Facebook

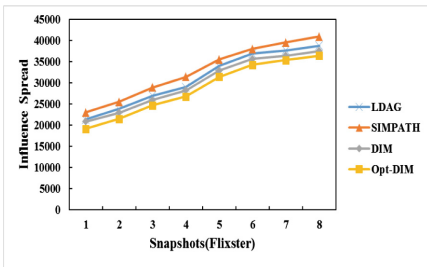


Fig. 8. Influence spread of Flixster dataset

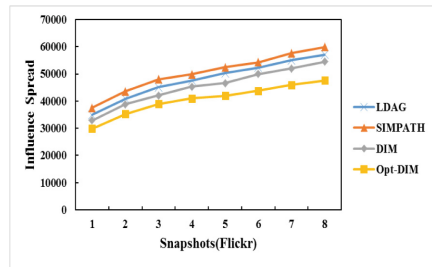


Fig. 9. Influence spread of Flickr dataset

6 Conclusions and Future Work

In this paper, we propose a dynamic influence maximization algorithm DIM, including two parts, Init_Seeding and Inc_SeedUpdate. Init_Seeding algorithm obtains the path set of all nodes by traversing the network graph, and computes the marginal influence of nodes according to the path set. Inc_SeedUpdate algorithm determines the influence spread change to incrementally update the seeds set when network changed. We also present two pruning strategies to further narrow the search space of influential nodes. The effectiveness and efficiency of DIM and Opt-DIM algorithm are verified by experiments.

In the future, we will study the probability distribution of edges and consider the influence maximization algorithm based on the probability distribution variation of edges.

Acknowledgment. This work was supported in part by the National Science Foundation of China (61632010, 61100048, 61370222), the Natural Science Foundation of Heilongjiang Province (F2016034), the Education Department of Heilongjiang Province (12531498).

References

1. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 61–70. ACM, Edmonton (2002)
2. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146. ACM, Washington D.C. (2003)
3. Leskovec, J., Krause, A., Guestrin, C.: Cost-effective outbreak detection in networks. In: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 420–429. ACM, San Jose (2007)
4. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1029–1038. ACM, Washington D.C. (2010)
5. Kim, J., Kim, S.K., Yu, H.: Scalable and parallelizable processing of influence maximization for large-scale social networks? In: 29th International Conference on Data Engineering, pp. 266–277. IEEE Computer Society, Washington, D.C. (2013)
6. Yang, Y., Mao, X., Pei, J., et al.: Continuous influence maximization: what discounts should we offer to social network users? In: 26th International Conference on Management of Data, pp. 727–741. ACM, San Francisco (2016)
7. Chen, W., Wei, L., Zhang, N.: Time-critical influence maximization in social networks with time-delayed diffusion process. *Chin. J. Eng. Des.* **19**(5), 340–344 (2015)
8. Zhuang, H., Sun, Y., Tang, J., et al.: Influence maximization in dynamic social networks. In: 13th International Conference on Data Mining, pp. 1313–1318. IEEE, Austin (2013)
9. You, Q., Hu, W., Wu, O.: Influence maximization in human-intervened social networks. In: 24th International Conference on Social Influence Analysis, pp. 9–14. IJCAI, Buenos Aires (2015)
10. Tong, G., Wu, W., Tang, S., et al.: Adaptive influence maximization in dynamic social networks. *IEEE/ACM Trans. Netw.* **99**, 112–125 (2015)
11. Liu, X., Liao, X., Li, S., et al.: On the shoulders of giants: incremental influence maximization in evolving social networks. *Comput. Sci.* (2015)
12. Goyal, A., Wei, L., Lakshmanan, L.V.S.: SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In: 11th International Conference on Data Mining, pp. 211–220. IEEE Computer Society, Washington, D.C. (2011)
13. Lu, Z., Fan, L., Wu, W., et al.: Efficient influence spread estimation for influence maximization under the linear threshold model. *Comput. Soc. Netw.* **1**(1), 1–19 (2014)