Influence Spreading Path and its Application to the Time Constrained Social Influence Maximization Problem and Beyond

Bo Liu, Gao Cong, Yifeng Zeng, Dong Xu, and Yeow Meng Chee

Abstract—Influence maximization is a fundamental research problem in social networks. Viral marketing, one of its applications, is to get a small number of users to adopt a product, which subsequently triggers a large cascade of further adoptions by utilizing “Word-of-Mouth” effect in social networks. Time plays an important role in the influence spread from one user to another and the time needed for a user to influence another varies. In this paper, we propose the time constrained influence maximization problem. We show that the problem is NP-hard, and prove the monotonicity and submodularity of the time constrained influence spread function. Based on this, we develop a greedy algorithm. To improve the algorithm scalability, we propose the concept of Influence Spreading Path in social networks and develop a set of new algorithms for the time constrained influence maximization problem. We further parallelize the algorithms for achieving more time savings. Additionally, we generalize the proposed algorithms for the conventional influence maximization problem without time constraints. All of the algorithms are evaluated over four public available datasets. The experimental results demonstrate the efficiency and effectiveness of the algorithms for both conventional influence maximization problem and its time constrained version.

Index Terms—Influence Spreading Path, Influence Maximization, Social Network, Large Scale, Time Constrained.

1 INTRODUCTION

The influence maximization problem has been extensively studied (e.g., [1]–[7], [9]). It aims to find a set of \( K \) influential nodes such that the expected number of nodes reached by influence spreading from the selected node set is maximized. Among others, a motivating application of influence maximization is viral marketing in social networks (e.g., Facebook), which has become a common ground for businesses to target potential customers. Viral marketing aims to select a small number of influential users to adopt a product, and subsequently trigger a large cascade of further adoptions by utilizing the “Word-of-Mouth” effect in social networks [10], [11]. For example, a pop vocal concert marketer may select a small number of influential users from a social network, and offer each of them a free ticket, such that the concert is widely known throughout the entire social network.

Recent research reveals that time plays an important role in the influence spread from one user to another [12] and the time needed for a user to influence another varies. Indeed, influence propagation time is considered in the recent work [12]–[16] on building the underlying influence propagation graph from real world log data.

On the other hand, in many real world viral marketing applications, people only care about how widely the influence is spread before a fixed time. For example, to market a pop vocal concert to be held on Sep 1st 2012, the marketer would want to maximize the number of users influenced before Sep 1st 2012. A conventional influence maximization model does not consider that influence among users may depend on the time. For example, some users may only pass the information to others after a rather long period. Consequently, the selected influential users may not spread the influence within a limited time. Indeed, users influenced after the concert would not bring any profit to the marketer. The conventional influence maximization solutions become invalid since the time is not considered in the influence propagation.

This calls for the problem of considering the influence maximization under the time constraint. We proceed to illustrate the idea of incorporating time factor in influence maximization using an example in Figure 1. In this example, five users are connected by five edges, each of which indicates a user may influence over another user. Numbers over each edge give the corresponding influencing probabilities, and the distribution of influencing delays. For example, user \( v_2 \) will influence \( v_5 \) with a probability of 0.7, and the influencing delay is distributed over the first two time units (e.g., day) with probability 5/7 and 2/7 respectively. This means that user \( v_2 \) would influence \( v_5 \) within the first time unit (resp. the second time unit) at a probability 0.7 * 5/7 (resp. 0.7 * 2/7), and \( v_2 \) cannot influence \( v_5 \) after the first two time units. Suppose we are asked to find a
single seed user to maximize the expected number of influenced users. Without any time constraint, user \( v_1 \) will be returned as the result. Intuitively, it is expected to influence the maximal number of users among all users. However, if we aim to find a single seed user that influences the maximal number of users within 1 time unit, user \( v_2 \) will become the new result. Intuitively, this is because \( v_1 \) can at most influence \( v_2 \) and \( v_3 \) in 1 time unit while \( v_2 \) influences \( v_4 \) and \( v_5 \) with a higher probability as given in Figure 1 (the algorithms for calculating the result will be presented in later sections).

In this paper, we define the time constrained influence maximization problem, which is based on the Latency Aware Independent Cascade influence propagation model, and which is shown to be \( NP\)-hard. We propose an algorithm that considers time factor in the process of Monte Carlo simulation to estimate the influence spread for a given seed set. This enables us to employ a greedy algorithm to solve the time constrained influence maximization problem. However, the greedy algorithm is computationally expensive particularly for solving a large scale of social networks. To facilitate the solutions, we propose the concept of Influence Spreading Path, based on which two methods for the time constrained influence maximization problem are designed. We further parallelize the algorithms for more efficiency improvement by exploiting the algorithmic independency. In addition, we generalize the proposed algorithms to solve the conventional influence maximization problem.

The contributions of this paper are summarized as follows:

- We define the time constrained influence maximization problem in social networks. We study the monotonicity and submodularity of the corresponding time constrained influence spread function. We propose a time step based simulation algorithm for estimating the time constrained influence. These lead to a simulation based approximate algorithm.
- We develop the logically augmented social networks and define an Influence Spreading Path for the time constrained influence maximization problem. Accordingly, we propose a set of more efficient algorithms that can be scaled to handle social networks of large scales. We design a parallel version of the proposed algorithms and show significant time savings.
- We generalize the Influence Spreading Path to solve the conventional influence maximization problem. The generalized algorithms perform better than the techniques for solving the conventional influence maximization problem.
- We demonstrate the algorithm performance over four public available datasets. The extensive experiments show that the Influence Spreading Path based algorithms outperform state-of-art techniques on solving both time and conventional influence maximization problems.

The remainder of this paper is organized as follows. The related work is reviewed in the next section. Section 3 presents a latency aware independent cascade model and the definition of time constrained influence maximization problem. In Section 4, we give a greedy algorithm for the time constrained influence maximization problem, and then propose a simulation and two Influence Spreading Path based solutions. In Section 5 we show that Influence Spreading Path can be used to solve conventional influence maximization problem. Section 6 presents the experimental study. Finally, Section 7 concludes this paper.

2 RELATED WORK

The problem of building the underlying influence propagation graph has been studied recently. Saito et al. [15] propose an asynchronous model to extend the traditional Independent Cascade Models by incorporating influence spreading delay information. The proposed asynchronous model is employed to facilitate model parameter learning of the influence graph. Other efforts of learning parameters of the influence graph from historical data include the work [12], [14]. The problem of building an influence graph is orthogonal to influence maximization problem, which assumes that the influence graph is known.

Richardson et al. [1], [2] are the first to study influence maximization problem in social networks. They formulate the problem with a probabilistic framework and employ Markov Random Field to solve it. Kempe et al. [3] formulate the problem as a discrete optimization problem, which is widely adopted by subsequent studies. They prove the influence maximization problem is \( NP\)-hard, and propose a greedy algorithm to approximately solve it by repeatedly selecting the node incurring the largest marginal influence increase to a seed set. To find the node incurring the largest marginal influence increase at each step, one needs to know influence spreads induced by different seed sets generated by adding each individual candidate node into current seed set.

However, the problem of calculating influence spread induced by a given seed set is very difficult (Chen et al. [5] prove it to be \#P-hard). Kempe et al. [3] propose to simulate influence spreading process starting from the given seed set for a large number of times, and then use the average value of simulation results to
approximate it. However, the simulation based method is computationally expensive and cannot scale well with large social networks [4]–[6]. To ease this problem, Leskovec et al. [4] propose a mechanism called Cost-Effective Lazy Forward (CELF) to reduce the number of times required to calculate influence spread, which will be used to optimize our algorithms in the conducted experiments. Chen et al. propose two fast heuristics algorithms, DegreeDiscount [5] and PMIA [6], to select nodes at each step of the greedy algorithm. At each step, DegreeDiscount adds the node with the largest degree to a seed set, and then degrees of neighbors of the selected node are discounted accordingly. PMIA calculates influence spread by employing local influence arborescences, which are based on the most probable influence path between two nodes. As PMIA needs to maintain arborescence for each node, it consumes a huge amount of memory, which makes it unsuitable to a large social graph. We compare with DegreeDiscount and PMIA in our experimental studies. In addition, Wang et al. [7] solve the problem by exploring the underlying community structure of social networks. Jiang et al. [8] employ the Simulated Annealing algorithm to find the top-k influential nodes from networks whose edges have the identical activation probability.

In parallel, Chen et al. [17] propose the time-critical influence maximization problem, in which the influencing model is a special case of the model proposed in this paper. In their model influence delays are constrained to follow the geometric distribution. In contrast, our model has no such a constraint and our algorithm is applicable when other distributions are used in the influencing model. Lee et al. [21] propose a different influence model where every active node has multiple chances to activate its neighbors, and the activation processes stop before a time.

3 Time Constrained Influence Maximization Problem

We present the conventional influence maximization problem and the Independent Cascade (IC) model in Section 3.1. Then, we present the proposed Latency Aware Independent Cascade (LAIC) model. Subsequently, we define the time constrained influence maximization problem in Section 3.2. Notations used in this paper are summarized in Table 1.

### 3.1 Conventional Influence Maximization

A conventional influence maximization problem aims to select $K$ nodes so that the expected number of nodes influenced by $K$ nodes will be maximized.

**Definition 1 (Influence Maximization Problem).** Given a social network $P = (\mathcal{V}, \mathcal{E})$, a positive integer $K < \mathcal{V}$, activating probability $\mathcal{T}_{uv} / (0,1)$ for each $(u,v) / \mathcal{E}$, find a seed set $S \rightarrow \mathcal{V}$ of $K$ nodes, such that the expected number of nodes influenced by $S$, $\sigma_T(S)$, is maximized.

### 3.2 Time Constrained Influence Maximization

The LAIC model considers the delayed influence propagation by encoding the time into the activation probability of edges in a social network. In the LAIC model, when a node $u$ is first activated at step $t$, it activates its currently inactive neighbor $v$ in step $t+\delta_t$ with probability $\mathcal{T}_{uv} \mathcal{T}_{uv}^{lat}(\delta_t)$, where $\delta_t$ is the influencing delay and is randomly drawn from the delay distribution $\mathcal{T}_{uv}^{lat}$. Note that a node can be activated at most once. If a node has been influenced by multiple neighbors, it is activated at
Theorem 1. Due to the limited space, we show in Theorem 1. The time constrained influence maximization problem is NP-hard.

4 Influence Spreading Path Based Solution

We present a greedy algorithm to calculate the expected influence spread in Section 4.1. To alleviate the computational complexity of the greedy algorithm, we propose a simulation based algorithm in Section 4.2, and define the Influence Spreading Path in Section 4.3. Subsequently, we develop an Influence Spreading Path based algorithm in Section 4.4. Furthermore, we improve the algorithm by employing faster marginal influence spread estimation in Section 4.5. In Section 4.6, we provide a parallelization version of the set of algorithms.

4.1 Monotonicity, Submodularity and Greedy Algorithm

Let $\sigma_T(S)$ be the expected number of nodes influenced by $S$ within $T$ time units. By replacing $\sigma(S)$ with $\sigma_T(S)$, we adapt the greedy algorithm [3] to approximately solve the time constrained influence maximization problem, which is given in Algorithm 1.

Algorithm 1: Greedy Algorithm Framework

Input: $G$, $T$, $K$, $P_{uv}$ and $P_{uv}^{\text{lat}}$

Output: $S$

1. initialize $S = \emptyset$
2. for $i \leftarrow 1 \text{ to } K$
3. $u \leftarrow \arg \max_v \sigma_T(S \cup \{v\}) - \sigma_T(S)$
4. $S \leftarrow S \cup \{u\}$
5. return $S$

The greedy algorithm repeatedly adds the node incurring the largest marginal influence increase to the seed set $S$, until $\vert S \vert = K$. The time complexity of Algorithm 1 is $O(KnV(\sigma_T(S)))$, where $n$ is the number of nodes in $P$ and $V(\sigma_T(S))$ the running time for calculating $\sigma_T(S \{ \{v\})$. As Theorem 2 shows the influence function $\sigma_T(S)$ is monotonous and submodular [19], and thus the greedy algorithm approximates the optimal solution with a lower bound ratio of $1 - 1/e$, where $e$ is the base of the natural logarithm [20].

Theorem 2. With the LAIC model, the influence function $\sigma_T(S)$ is monotonous and submodular.

The main difficulty in applying the greedy algorithm lies in calculating the expected influence spread for a given set of seeds (Line 3 of Algorithm 1), whose special case has been shown to be #P-hard [6]. In the following sections, we propose a set of approximate algorithms including a simulation based algorithm and two Influence Spreading Path based algorithms.

4.2 Simulation based Algorithm for $\sigma_T(S)$

We propose Algorithm 2 to simulate the time constrained influence spreading process based on time steps. Note that Algorithm 2 differs from the simulation algorithm for conventional influence maximization problem [3], which is based on Breadth-first Search (BFS) and does not consider time factor.

Algorithm 2: $\sigma_T(S)$ based on Simulation

Input: $G$, $T$, $S$, $P_{uv}$ and $P_{uv}^{\text{lat}}$

Output: $\sigma_T(S)$

1. $v.\text{status} \leftarrow \text{inactive}$, $v.\text{actTime} \leftarrow +\infty$ for $v \in V \setminus S$
2. $v.\text{status} \leftarrow \text{active}$, $v.\text{actTime} \leftarrow 0$ for $v \in S$
3. $A_0 \leftarrow S$
4. $t \leftarrow 1$
5. do
6. for $u \in A_{t-1}$ do
7. if $u.\text{status} \neq \text{active}$ do
8. draw $\text{flag}$ from Bernoulli($P_{uv}$)
9. if $\text{flag} = 1$ then
10. draw $\delta_t$ from $P_{uv}^{\text{lat}}$
11. if $u.\text{status} = \text{inactive}$ then
12. $v.\text{status} \leftarrow \text{latent active}$
13. $v.\text{actTime} \leftarrow t + \delta_t$
14. else if $t + \delta_t < v.\text{actTime}$ then
15. $v.\text{actTime} \leftarrow t + \delta_t$
16. $A_t \leftarrow \{u | u.\text{actTime} = t \land u.\text{status} = \text{latent active}\}$
17. $u.\text{status} \leftarrow \text{active}$ for $u \in A_{t-1}$
18. $t \leftarrow t + 1$
19. while $\vert \{u | u.\text{status} = \text{latent active}\} \neq 0$ or $A_t \neq \emptyset$;
20. return $\sum_{j=0}^{1} \vert A_j \vert$

In Algorithm 2, we simulate the influence propagation process starting from $S$. In the beginning, all nodes in $S$ are set to be active, while all other nodes are set to be inactive (Lines 1-2 of Algorithm 2). The set of nodes activated at time $t$ are denoted by $A_t$. Nodes in $S$ are treated as being activated at time 0 (Line 3). At time $t > 0$, each node $u / A_{t-1}$ intends to activate each of its inactive or latent active (to be explained) outgoing neighbors $v / N_{\text{out}}(u)$ with the probability $P_{uv}$. If $u$ successfully activates $v$ (Lines 9-20), an activating latency $\delta_t$ ($\delta_t = 0, 1, 2...$) is drawn from the discrete distribution.
Algorithm for calculating the activation probability of a fluence spreading path is given. Then we propose an algorithm to activate nodes and newly activated nodes. When the process terminates, the number of activated nodes is returned (Line 27).

**Time and space complexities** Let \( n \) (resp. \( m \)) be the number of nodes (resp. edges) in social network \( P \). The first four lines of Algorithm 2 take \( O(n) \) time. For the entire while loop, the dominant cost is on exploring the graph starting from \( S \) along edges. In the worst case, the algorithm needs to explore all nodes and edges in the graph. Thus the running time is \( O(n + m) \) for the while loop, which is also the time complexity of Algorithm 2.

In addition to the input social graph, Algorithm 2 only needs to store \( status \) and \( actTime \) for each node, the space needed by which is \( O(n) \). Thus the space complexity of Algorithm 2 is \( O(n + m) \), which is dominated by the input of the social network.

To approximate the expected influence spread within \( T \) time units, we may repeat Algorithm 2 for a large number (\( R \)) of times and average the returned numbers. Consequently the total running time of the combination of Algorithm 1 and 2 is \( O(KnR(n+m)) \). By following [3], [5], [6], \( R = 20,000 \) simulations are employed to calculate the expected influence spread for a given seed set.

### 4.3 Influence Spreading Path based Activation Probability Calculation

Due to the computational curse, the simulation based algorithm is not suitable to large social networks. We proceed to describe how a social network is augmented into an influence network using the social network structure, we logically augment the original social network \( P = (U, G) \) into a directed multigraph \( G_T = (V, E) \), where \( V = U \). For each \( (u, v) \in G \), we put \( T \) edges, \( e_{uv}^1, e_{uv}^2, \ldots, e_{uv}^T \) from \( u \) to \( v \) in \( G \). Each edge \( e_{uv}(\in E) \) is guarded with two values, i.e., \( length(e_{uv}) = t \) and \( prob(e_{uv}) = \tau_{uv}^T \mu^t(T) \).

Figure 2 gives the multigraph augmented from the example of social network in Figure 1 under the case of \( T = 2 \). We note that this augmentation is done logically. All algorithms proposed in this paper are able to infer the augmented graph from an original graph on the fly.

#### 4.3.2 Constrained Influence Spreading Path

Given a seed set \( S \), the expected influence spread within time \( T \), \( \sigma_T(S) \), is the expected number of nodes activated no later than time \( T \), denoted by \( \sum_{u \in V} \mathcal{E}T_T(u, S) \), where \( \mathcal{E}T_T(u, S) \) is the probability that \( S \) activates \( u \) within \( T \). It is easy to find out that \( \mathcal{E}T_T(u, S) = 0 \), if there is no path from \( S \) to \( u \) in the augmented directed multigraph \( G_T = (V, E) \). Thus in what follows, we ignore those nodes not reachable from \( S \).

To estimate \( \mathcal{E}T_T(u, S) \) for each node \( u \), we define Influence Spreading Path in the augmented graph below.

**Definition 3** (Influence Spreading Path). Given a seed set \( S \) and a directed multigraph \( G = (V, E) \), a simple path \( p = (u_1, e_1 \in E, u_2 \in E, \ldots, u_k \in E) \) in graph \( G \) is an Influence Spreading Path, if and only if \( u_1 / S \) and \( u_k / S \) for \( i \neq 1 \), where \( k > 1 \). For an influence spreading path \( p \), the length of \( p \) is \( \sum_{i=1}^{k-1} length(e_i) \), while the probability of \( p \) is \( \prod_{i=1}^{k-1} prob(e_i) \).

From Definition 3, we notice that an Influence Spreading Path cannot contain duplicate nodes, as a node cannot be activated more than once. Furthermore, except the starting point, an influence spreading path cannot contain any of other nodes belonging to \( S \), which resides in the fact that seed nodes are already in active state at the very beginning and cannot be activated at a later step. Note that the proposed algorithms do not need the detailed path information, and we only need to store length, probability and the ending node of each Influence Spreading Path.

We observe that each Influence Spreading Path \( p \) ending with \( u \) gives a possible way for \( S \) to activate \( u \). The activating time taken by following \( p \) to activate \( u \) is \( length(p) \), while the activating probability of this path is \( prob(p) \). For a given seed set \( S \), we denote \( ISP(u, S) \) to be all possible influence spreading paths ending with \( u \). Note that \( ISP(u, S) \) grows exponentially as the number of nodes increases. To reduce the number of paths in \( ISP(u, S) \), we apply two restrictions to filter out some Influence Spreading Paths which are not or less related to our problem. First, we prune paths with length larger than \( T \), which are not related to influence spread within
time $T$. Furthermore, we filter out paths with probability less than a small threshold $\theta$, as Influence Spreading Paths with small probabilities have limited impact on the influence spread estimation. The resulting constrained Influence Spreading Paths are denoted by $ISP_{\theta,T}(u,S)$.

### 4.3.3 Activation Probability Calculation based on Influence Spreading Paths

By assuming all Influence Spreading Paths ending at $u$ ($ISP_{\theta,T}(u,S)$) are independent with each other, we are able to calculate the probability $u$ gets activated by $S$ within time $T$ ($E_T^{(t)}(u,S)$) from $ISP_{\theta,T}(u,S)$. The computation is outlined in Function $AP$. The function iterates over all possible time steps from 1 to $T$, calculates the probability that $u$ is first activated at time $t$ ($E_T^{(t)}(u,S)$) (Line 3), and adds it to $E_T(u,S)$ at Line 4. At Line 3, $E_T(u,S)$ is the probability $u$ has not been activated before $t$, and $1 - \prod_{p \in ISP_{\theta,T}(u,S),\text{length}(p)=t}(1 - \text{prob}(p))$ is the probability $u$ is activated at time $t$. At the end of each iteration $t$, $E_T^{(t)}(u,S)$ is updated to store the probability $u$ is activated before $t + 1$. The loop results in the probability $E_T(u,S)$ that $u$ is activated within time $T$.

#### Time and space complexities

For the running time, the dominant part of Function $AP$ is the for loop in which every Influence Spreading Path in $ISP_{\theta,T}(u,S)$ is checked exactly once. Thus the running time of Function $AP$ is $O(|ISP_{\theta,T}(u,S)|)$. Note that the space complexity of Function $AP$ is also $O(|ISP_{\theta,T}(u,S)|)$.

**Function AP**

**Input:** $ISP_{\theta,T}(u,S)$, $T$

**Output:** $AP_T(u,S)$

1. $AP_T(u,S) \leftarrow 0$
2. for $t \leftarrow 1$ to $T$ do
   3. $AP_T^{(t)}(u,S) \leftarrow (1 - AP_T(u,S))(1 - \prod_{p \in ISP_{\theta,T}(u,S),\text{length}(p)=t}(1 - \text{prob}(p)))$
   4. $AP_T(u,S) \leftarrow AP_T(u,S) + AP_T^{(t)}(u,S)$
5. return $AP_T(u,S)$

#### 4.4 Influence Spreading Path based Algorithm for $\sigma_T(S)$

Algorithm 3 computes the expected influence spread within time $T$ for a given seed set ($\sigma_T(S)$). First, Algorithm 3 gets all constrained Influence Spreading Paths starting from $S$ by a Depth-First Search (DFS) (Line 2), which are then divided into disjoint sets based on their ending nodes (Line 3). For each node $u$ with at least one constrained Influence Spreading Path, i.e., $ISP_{\theta,T}(u,S) \neq \emptyset$, Function $AP$ is applied to calculate the probability $E_T(u,S)$ that $u$ is activated by $S$ within time $T$ (Line 5). Finally, activation probabilities of all nodes are summed together and returned as the expected influence spread of $S$.

Similar to Algorithm 2, Algorithm 3 is embedded in Algorithm 1 (calculating $\sigma_T(S)$) to find a seed set of $K$ nodes.

---

**Algorithm 3: $\sigma_T(S)$ based on Influence Spreading Path**

**Input:** $G$, $\theta$, $T$, $S$

**Output:** $\sigma_T(S)$

1. $\sigma_T(S) \leftarrow 0$
2. get all Influence Spreading Paths with length no larger than $T$ and probability no less than $\theta$ by DFS.
3. divide them into different $ISP_{\theta,T}(u,S)$.
4. for every $u$ with non-empty $ISP_{\theta,T}(u,S)$ do
   5. $\sigma_T(S) \leftarrow \sigma_T(S) + AP(ISP_{\theta,T}(u,S),T)$
5. return $\sigma_T(S)$

#### Time and space complexities

Let $n_{\theta T} = \max_{|S| \le K} |ISP_{\theta,T}(S)| \approx |ISP_{\theta,T}(S)|$, where $|ISP_{\theta,T}(S)|$ be the number of Influence Spreading Paths starting from $S$ with length no less than $T$ and probability no less than $\theta$. The second line of Algorithm 3 can be done using DFS algorithm in $O(n_{\theta T})$ time, which is also the time needed for the third line. As calculating $E_T(u,S)$ by Function $AP$ takes $O(|ISP_{\theta,T}(u,S)|)$ time and $\sum_{u \in V} |ISP_{\theta,T}(u,S)| \approx |ISP_{\theta,T}(S)| \approx n_{\theta T}$, the for loop also takes $O(n_{\theta T})$ time. Thus the total running time of Algorithm 3 is $O(n_{\theta T})$.

Note that the Influence Spreading Path based solution (combination of Algorithms 1 and 3) takes $O(Kn_{\theta T})$ time, which is much less than the time needed by the simulation based solution (combination of Algorithms 1 and 2) $O(KnR(n + m))$. It is obvious to see that the space complexity of Algorithm 3 is $O(n + m + n_{\theta T})$, where the $n + m$ is the size of social graph, and $n_{\theta T}$ for storing $ISP_{\theta,T}(S)$.

---

### 4.5 Faster Marginal Influence Spread Estimation

In the greedy Algorithm 1, when trying to add one more node into the currently selected seed set $S$, we need to calculate the marginal influence increase brought by adding each $u / V \backslash S$. Instead of calculating $\sigma_T(\{u\})$ from scratch in Algorithm 3, we propose to employ a faster marginal influence spread estimation.

Suppose the currently selected seed set is $S$, we want to calculate the marginal influence spread increase if node $v$ is added to $S$, i.e., $\sigma_T(S \cup \{v\}) - \sigma_T(S)$, which is obviously no larger than $\sigma_T(\{v\})$. As $\sigma_T(\{v\})$ is already known in the computation for selecting the first seed node, we propose to approximate $\sigma_T(S \cup \{v\}) - \sigma_T(S)$ by making a discount of $\sigma_T(\{v\})$ in Equation 1.

$$
\sigma_T(S \cup \{v\}) - \sigma_T(S) \approx \frac{\sum_{(u,w) \in E} T_{uw}(1 - T_{Sw})\sigma_T(w)}{\sum_{(u,w) \in E} T_{uw}\sigma_T(w)}
$$

(1)

where $T_{Sw} = 1 - \prod_{(u,w) \in E, \text{seed } u \in S}(1 - T_{uw})$ if $w / V \backslash S$; otherwise, $T_{Sw} = 0$. In other words, $T_{Sw}$ is the probability $w$ gets immediately activated by seed nodes. The rationality behind Equation 1 is that the marginal influence increase is a discount of $\sigma_T(\{v\})$. The higher probability $v$’s neighbors are already activated by $S$, the larger discount should be applied to $\sigma_T(\{v\})$. With
this marginal influence spread increase approximation, we propose Algorithm 4 to solve the time constrained influence maximization problem.

Algorithm 4: Marginal Discount of Influence Spread Path

Input: $G, T, K, P_{uw}, P_{uv}^{lat}, \theta$
Output: $S$
1 for every $u \in V$ do
  2 calculate $\sigma_T(\{u\})$ by Algorithm 3.
  3 $u \leftarrow \arg \max_v \sigma_T(\{v\})$
  4 $S \leftarrow \{u\}$
  5 $P_{Sw} \leftarrow P_{uw}$ for $w \in N_{out}(u)$
  6 $P_{Sw} \leftarrow 0$ for $w \notin N_{out}(u)$
  7 for $k \leftarrow 1$ to $K - 1$ do
    8 $u \leftarrow \arg \max_{v,w} \sigma_T(\{v\}) \frac{\sum_{(v,w) \in E} P_{uv} (1 - P_{Sw}) \sigma_T(\{w\})}{\sum_{(v,w) \in E} P_{uv} \sigma_T(\{w\})}$
    9 $S \leftarrow S \cup \{u\}$
  10 update $P_{Sw}$ for every $w \in N_{out}(S)$.
11 return $S$

Algorithm 4 calculates time constrained influence spread based on influence spreading paths for each single node (Lines 1-3). Seed nodes are selected by picking the node with the largest discounted marginal influence one by one (Lines 8-12).

**Time and space complexities** As Algorithm 3 takes $O(nT)$ time, the first for loop of Algorithm 4 takes $O(nmT)$ time. Line 9 takes $O(ne_{max})$ time while line 11 takes $O(Ke_{max})$ time, where $e_{max}$ is the largest degree among all nodes. Thus the second for loop takes $O((K - 1)e_{max})$ time, and the total running time of Algorithm 4 is $O(n(nT) + (K - 1)e_{max})$. Note that Algorithm 4 itself solves the time constrained influence maximization problem, and does not need to be combined with Algorithm 1. By comparing Algorithm 4 and the combination of Algorithms 1 and 3, whose running time is $O(KnmT)$, we find that they have the same running time when $K = 1$, and Algorithm 4 runs faster when $K > 1$. This observation is consistent with the experimental results that will be presented in the experimental section. The memory space needed by Algorithm 4 is dominated by running Algorithm 3 at line 2, and thus the space complexity for Algorithm 4 is the same as that for Algorithm 3, which is $O(n + m + nT)$.

### 4.6 Parallelized Algorithm

The running time of Algorithm 4 (resp. the combination of Algorithms 1 and 3) is mostly dominated by applying Algorithm 3 to calculate $\sigma(\{v\})$ for each $v \in V$ in Line 1-3 of Algorithm 4 (resp. Line 3 of Algorithms 1). Different from local arborescences based methods [6], [17], Algorithm 3 is based on Influence Spreading Path, in which there exists no inter-dependency between calculating $\sigma(\{v\})$ for different node $v \in V$. Therefore, the most time consuming parts of Algorithm 4 (resp. the combination of Algorithms 1 and 3) can be easily parallelized on a multi-core or distributed system with a multi-threaded Queue or distributed Queue.

As depicted in Figure 3, all nodes $\cup$ are put into the Queue, every thread repeatedly fetched node $v$ from the Queue and applies Algorithm 3 to calculate $\sigma(v)$. Let $c$ be the total number of cores on a single machine or in a distributed system, parallelized Influence Spreading Path based methods can run up to $c$ times faster. The effectiveness of this parallelization strategy will be demonstrated in the experimental study.

### 5 Applying Influence Spreading Path to Conventional Influence Maximization Problem

The conventional influence maximization problem based on IC Model [18] can be regarded as a special case of the time constrained influence maximization problem. The proposed Influence Spreading Path based methods can be applied to the conventional influence maximization problem with slight modifications on Function $ET(u, S)$ and Algorithm 1, 3, 4. In what follows we brief the modifications.

Function $ET(u, S)$ is modified into $ET(u, S)$, which is presented below. It takes as input the Influence Spreading Paths starting from $S$ and ending with $u$ without considering time constraint $T$; it returns the probability $u$ gets influenced by seed set $S$, which can be computed under the same assumption made for time constrained problem previously. With the assumption that all Influence Spreading Paths starting from $S$ and ending with $u$ are independent, we can calculate $ET(u, S)$ by iterating over all $p / ISPs_{\theta}(u, S)$ as described in Lines 2-4 of Function $AP$. $ET(u, S)$ at the right hand side of Line 3 is the probability $u$ gets influenced by following the paths which has been checked by the for loop before current iteration. $(1 - ET(u, S))prob(p)$ is the probability $u$ is not influenced by previously checked paths and influenced by current path $p$.

**Time and space complexities** For the running time, the dominant part of Function $AP$ is the for loop in which every Influence Spreading Path in $ISPS_{\theta}(u, S)$ is checked exactly once. Thus the time complexity of Function $AP$ is $O(ISPS_{\theta}(u, S))$. The space complexity of Function $AP$ is also $O(ISPS_{\theta}(u, S))$.

Algorithm 1 is modified into Algorithm 5, where $\sigma(S)$ is computed by Algorithm 6.
Algorithm 6: Greedy Algorithm Framework (For Conventional Influence Maximization Problem)

Function AP (For Conventional Influence Maximization Problem)

Input: ISPθ(u, S)
Output: AP(u, S)
1 AP(u, S) ← 0
2 for p ∈ ISPθ(u, S) do
3     AP(u, S) ← AP(u, S) + (1 − AP(u, S))prob(p)
4 return AP(u, S)

Algorithm 5: Greedy Algorithm Framework (For Conventional Influence Maximization Problem)

Input: G, K and Puv
Output: S
1 initialize S = ∅
2 for i ← 1 to K do
3     u ← arg maxv ∈ S ∪ {v} σ(S ∪ {v}) − σ(S)
4     S ← S ∪ {u}
5 return S

Algorithm 6: σ(S) based on Influence Spreading Path (For Conventional Influence Maximization Problem)

Input: G, θ and S
Output: σ(S)
1 σ(S) ← 0
2 get all Influence Spreading Paths with probability no less than θ by DFS.
3 divide them into different ISPθ(u, S).
4 for every u with non-empty ISPθ(u, S) do
5     σ(S) ← σ(S) + AP(ISPθ(u, S))
6 return σ(S)

To modify Algorithm 4 to cope with the conventional influence maximization problem, we replace all σT(S) with σ(S), which can be computed by Algorithm 6.

We note that the parallelization strategy in Section 4.6 is also applicable to Influence Spreading Path based methods for conventional influence maximization problem.

6 EXPERIMENTS

6.1 Experimental Setup

Datasets Four public real-world social networks 1 are used in the experiments, which are also widely used in previous work on influence maximization. The basic statistics of these networks are summarized in Table 2. The first one (Wiki) is a Wikipedia voting network where nodes represent Wikipedia user and an edge from node i to j represents that user i voted on user j. The second one (Epinions) is a who-trust-whom social network of a general consumer review site Epinions.com. The third one (Slashdot) is a social network extracted from the user community of Slashdot.org. The last one (LiveJournal) is a large social network formed by LiveJournal community.

<table>
<thead>
<tr>
<th>Networks</th>
<th>Wiki</th>
<th>Epinions</th>
<th>Slashdot</th>
<th>LiveJournal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Number</td>
<td>7,115</td>
<td>75K</td>
<td>82K</td>
<td>4.8M</td>
</tr>
<tr>
<td>Edge Number</td>
<td>2283</td>
<td>506K</td>
<td>948K</td>
<td>68.9M</td>
</tr>
<tr>
<td>Clustering Coefficient</td>
<td>0.2089</td>
<td>0.2293</td>
<td>0.0617</td>
<td>0.3121</td>
</tr>
</tbody>
</table>

Evaluated Methods The experimental study is to demonstrate the capability of Influence Spreading Path based methods for solving both conventional and time constrained influence maximization problems. We note that all methods proposed in this paper are based on the greedy algorithm framework. The difference lies in the way of calculating the marginal influence increase, i.e., Line 3 of Algorithm 1. The following methods are evaluated.

- Monte Carlo (MC). For time constrained influence maximization problem, MC calculates both σT(S \ {v}) and σT(S) by simulations (combination of Algorithms 1 and 2). For conventional influence maximization, MC is the simulation based greedy algorithm proposed in [3]. 20,000 simulations are employed for each seed set by following [3], [5], [6].
- Influence Spreading Path (ISP). Calculate both σT(S \ {v}) and σT(S) by using Influence Spreading Paths (combination of Algorithms 1 and 3). The Influence Spreading Paths starting from each seed set are calculated from scratch by DFS.
- Marginal Discount of Influence Spread Path (MISP). Calculate influence spread σT(u) for each single node u with Influence Spreading Paths starting from u, then select a seed node with the largest discounted marginal influence spread one by one (Algorithm 4).
- Random. Randomly select K nodes as seeds, which acts as the baseline method.
- Degree Discount (DC). The degree discount heuristic proposed by [5].
- Prefix excluding Maximum Influence Arborescence (PMIA). PMIA [6] is a state-of-the-art solution for

conventional influence maximization problems.

- Maximum Influence Arborescence for IC-M (MIAM) and Maximum Influence Arborescence with Converted propagation probabilities (MIAC) [17]. MIAM and MIAC are proposed for time constrained influence maximization problem based on PMIA algorithm. They only apply to the scenario of geometric influencing delay (to be further explained in the next subsection).

The implementations of DC, PMIA, MIAM and MIAC are provided by their authors. Note that all evaluated methods are enhanced by CELF [4] optimization if applicable.

We apply the aforementioned algorithms to the time constrained influence maximization problem, and all except MIAM and MIAC to the conventional influence maximization problem, for which MIAM and MIAC reduce to PMIA.

Parameter Setting The activating probability $T_{uv}$ of each edge $(u, v)$ is set by the “Weighted Cascade” policy, which is widely adopted by the existing conventional influence maximization techniques [3], [5], [6]. With “Weighted Cascade” policy, $T_{uv}$ is set to be $\frac{1}{N_{in}(v)}$, where $N_{in}(v)$ is the indegree of $v$.

In time constrained influence maximization problems, we consider two types of distributions for the influencing delays ($T^{iol}$), namely Poisson Delay Distribution or Geometric Delay Distribution. For each node $u / v$, the parameter for its Poisson distribution (expected number of occurrences in a given interval) is randomly selected from the set $\{1, 2, 3, ..., 20\}$; the parameter for its Geometric distribution is generated by $5/(d^{out}(u)+5)$, which follows the same way as [17]. We note that the distributions of both activating probability and influencing delay are orthogonal to the proposed methods.

The threshold parameters for PMIA, MIAM and MIAC are set to $\frac{1}{20}$ suggested by [6], [17]. We ran them with other threshold values, which resulted in less influence spread.

Parameter $\theta$ controls the number of Influence Spreading Paths for MISP and ISP. Intuitively, a smaller value of $\theta$ results in a larger number of Influence Spreading Paths used by MISP and ISP, and thus should achieve larger influence spread. However, on the other hand, a smaller value of $\theta$ incurs a larger amount of running time. Thus there exists a tradeoff between influence spread and running time, which is tunable by $\theta$.

To investigate the tradeoff and select an optimal value of $\theta$, we ran MISP and ISP with different values of $\theta$ for both conventional and time constrained problems. The running time and influence spread for different $\theta$ on Wiki dataset with $T = 10$ (for time constrained problem only), $K = 50$ are depicted in Figures 4 and 5. Note that results for other datasets and/or different values of $T$ and $K$ are similar, which are not included in this paper due to the limited space. Not surprisingly, Figures 4 and 5 show that a smaller value of $\theta$ achieves larger influence spread but consumes more running time for both MISP and ISP methods. As both MISP and ISP achieve relatively large influence spread and short running time for time constrained (resp. conventional) influence maximization problem with $\theta = 10^{-5}$ ($N_{in}(v)$), $\theta$ is set to $10^{-5}$ ($\frac{1}{20}$) for time constrained (resp. conventional) problem in the rest of the experimentations.

Measurement For the time constrained social influence maximization problem, a critical performance metric is the number of nodes influenced by the selected seed set within a given time. As the time constrained influence maximization problem is NP-hard, we are not able to get the result in polynomial time. Thus we apply 20,000 Monte Carlo simulations with seed set selected by each evaluated method, and the average influenced node number is used as the influence spread of the seed set. We also measure the running time and memory needed for each method. Furthermore, we will analyze the impact of different values of $T$ on the time constrained influence maximization problem.

For the conventional influence maximization problem, we measure the number of nodes influenced by the selected seed set, which is calculated by applying 20,000 Monte Carlo simulations as done by the previous work. Similarly we also measure the running time and memory needed for each method.
All algorithms are implemented in C++ language, and compiled by gcc 4.4.3 on a Linux server with an 8-core Intel Xeon 3.0 GHz CPU and 12 GB memory.

6.2 Experimental Results

In this section, we present the experimental results of the proposed methods on four real world social networks.

6.2.1 Influence Spread

**Time constrained influence maximization problem** MIAM and MIAC are only applicable to Geometric influence delay distribution. All the other six methods outlined in Section 6.1 are evaluated over datasets Wiki, Epinions, and Slashdot for both Geometric and Poisson distributions. However, PMIA, MIAM and MIAC are not evaluated on the LiveJournal dataset as the memory needed for these methods exceeds 12GB, which is the total amount of available memory in the experiments. We cannot obtain the result for MC method on LiveJournal dataset after running it for two days.

Figure 6 shows the results of influence spread over the four datasets with $T = 10$ for different $K$ values. It shows that both ISP and MISP methods achieve similar influence spread as the computationally expensive greedy algorithm MC, which verifies the effectiveness of Influence Spreading Path based methods.

As to MIAC and MIAM, which are developed for the time constrained problem with Geometric influence delay distribution, MIAC achieves less influence spread than ISP and MISP. Though MIAM is able to achieve similar influence spread as ISP and MISP, it cannot run with LiveJournal due to a huge memory consumption. As expected, a larger number of seed nodes achieve larger influence spread for all evaluated methods, and the randomly selected seed set result in very poor performance.

Among the algorithms originally designed for conventional influence maximization problem, PMIA performs the best, but it achieves considerably lower influence spread than do MISP, ISP and MC, which demonstrates that methods for conventional influence maximization problem do not work for the time constrained version.

**Conventional influence maximization problem** Figure 7 depicts the influence spread generated by different methods for conventional influence maximization problems. Again, MC and PMIA cannot run with LiveJournal due to either long running time or huge memory consumption. From Figure 7, we can see that ISP and MISP achieve similar influence spread as computationally ex-
pensive MC. Random and DC generate low influence spread. PMIA is able to achieve similar influence spread as ISP and MISP, but the huge memory consumption limits its applicability to large datasets such as LiveJournal.

### 6.2.2 Running Time and Memory Usage

**Poisson Delay Distribution**

- Fig. 8. Running time on four real world social networks for different values of $K$ ($T = 10$, Time Constrained Version).

**Geometric Delay Distribution**

- (a) Wiki
- (b) Epinions
- (c) Slashdot
- (d) LiveJournal

**Conventional influence maximization problem**

Figure 9 shows the running time for different methods. Again, PMIA cannot run with LiveJournal, MC cannot finish over Epinions with $K > 20$, and Slashdot with $K > 10$ in two days, which is the reason why we do not have the corresponding data points in Figure 9. We observe that MISP consistently runs faster than other methods. Again, the running time of methods other than MISP increases as $K$ increases, while that of MISP almost remains constant.

**Memory Usage**

Tables 4, 5 and 6 show the peak memory usage of each method for different datasets with $T = 10$ for conventional and time constrained influence maximization problems, respectively. We find that Random, MISP and MC always need the same amount of memory, which is mainly occupied by the social network data. For the two Influence Spreading Path based methods, the memory consumption of ISP grows as $K$ increases, while the memory needed by MISP remains constant. PMIA, MIAM and MIAC consume the largest amount of memory, which renders them inapplicable to social networks of large scales (e.g., LiveJournal).

### 6.2.3 Effect of Different Values of $T$

To investigate the impact of $T$ on the algorithm performance, we run MC with Wiki, Slashdot and Epinions datasets for $T = \{1, 2, ..., 10\}$ (as indicated in the previous sections, we cannot run MC with LiveJournal). Tables 7 and 8 depict the overlaps of seed sets returned by MC for different values of $T$ with $K = 50$ for Poisson and Geometric distributions respectively. For example, value 34 at row $T = 1$ and column $T = 4$ in Table 7 (Wiki) is the number of common nodes for the two T values.
that time constraint plays an important role in influence maximization problem, and the set of nodes maximizing influence spread before a given time do not necessarily maximize that for a different time constraint.

To investigate how the value of $T$ affects the running time needed and influence spread achieved by different methods, we show the running time and influence spread for different $T$ on Wiki dataset with $K = 50$ in Figure 10. Note that results for other datasets and/or different values of $K$ are similar, which are not included in this paper due to the limited space. As the running time for Random and DC is trivial, to make the figure more distinguishable, Random and DC methods are excluded from these figures. We find that the running time of MISP, ISP, MIAM and MC increases as $T$ increases, while that of PMIA and MIAC remains constant. MISP achieves much less running time than MC and ISP. MIAC runs faster than all other methods, but it achieves less influence spread and consumes a huge amount of memory. Again, MC needs the largest amount of running time among all methods. We also find that all methods achieve more influence spread as $T$ increases. This is due to the fact that a larger value of $T$ poses less restriction on time slots during which influence spread is counted. Again, MC, ISP, MISP and MIAM achieve

TABLE 3
Numbers of ISP Lengths over Four Datasets

<table>
<thead>
<tr>
<th>Length</th>
<th>Wiki</th>
<th>Epinions</th>
<th>Slashdot</th>
<th>LiveJournal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>2</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>3</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>4</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>5</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>6</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>7</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>8</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>9</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>10</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
</tbody>
</table>

TABLE 5
Memory Usage in MB ($T = 10$, Poisson Delay Distribution)

<table>
<thead>
<tr>
<th>Length</th>
<th>Wiki</th>
<th>Epinions</th>
<th>Slashdot</th>
<th>LiveJournal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>2</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>3</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>4</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>5</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>6</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>7</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>8</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>9</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
<tr>
<td>10</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
<td>5785</td>
</tr>
</tbody>
</table>

We find that seed sets maximizing influence spread with different time constraints differ significantly. We argue that time constraint plays an important role in influence spread.

Fig. 9. Running time on four real world social networks for different values of $K$ (Conventional Version).

Fig. 10. Results on Wiki with different $T$ ($K = 50$).
similar influence spread, which is much more than that of other methods.

6.2.4 Effect of Parallelized Processing

Parallelized algorithms are tested in the same environment as described in Section 6.1. Figure 11 shows the running time needed by ISP and MISP over LiveJournal with different number of threads running on different CPU cores. As expected, the running time decreases as more threads are added. This demonstrates that the parallelism of ISP and MISP further speeds up the solutions.

6.3 Summary and Discussion

From the experimental results, we find that time constraint plays an important role in influence maximization problem. Straightforward methods, like Random and DC, are not suitable for the time constrained influence maximization problem thereby leading to poor influence spread. PMIA, a state-of-the-art solution for the conventional influence maximization problem, achieves much less time constrained influence spread than do DC, ISP and MISP. Another drawback of applying PMIA to maximize time constrained influence is its large memory consumption, which makes it unsuitable for large social networks. MIAM is able to achieve similar influence spread as ISP and MISP, while MIAC performs worse in terms of influence spread when Geometric delay distribution is employed. MIAM and MIAC suffer from the same problem as PMIA on large networks, i.e., huge memory consumption. By investigating the effect of different values of $K$, we find that the set of nodes necessarily maximize that for a different time constraint, which shows that time constraint plays an important role in influence maximization problem. Influence Spreading...
Path based methods (ISP and MISP) run much faster than other methods and can be easily parallelized.

Influence Spreading Path based method ISP and MISP can be successfully used to solve conventional influence maximization problem. Moreover, MISP runs faster than the state-of-the-art method PMIA, achieves similar influence spread, and needs much less memory.

One limitation of Influence Spreading Path based methods is that parameter $\theta$ needs to be manually tuned to make a good tradeoff between influence spread and running time.

7 Conclusion

In this paper, we define a new problem of the time constrained influence maximization in social networks based on a Latency Aware Independent Cascade model. We develop a simulation based greedy algorithm with performance guarantees to solve the problem. However, the simulation based implementation of the greedy algorithm is rather expensive, and is not scalable for large social networks. We propose to use Influence Spreading Paths to quickly and effectively approximate the time constrained influence spread for a given seed set, which is the expensive part of the greedy algorithm. Further, by employing faster marginal influence spread calculating methods, we propose MISP to improve the speed of ISP. Experimental results show that MISP is the fastest and multiple orders of magnitude faster than simulation based greedy algorithm MC while achieving similar time constrained influence spread. Other nice properties of MISP include that its running time almost remains constant as $K$ increases, and can be easily parallelized.

Influence Spreading Path based methods are also successfully applied to conventional influence maximization problem. Experimental results show that MISP outperforms the state-of-the-art method PMIA in terms of running time and memory consumption, while achieving similar amount of influence spread.

References


Bo Liu is working with Facebook Inc. He received his PhD in 2009 from Huazhong University of Science and Technology, China. Before he joined Facebook, Dr. Liu was a lecturer in Huazhong University of Science and Technology and research fellow in Nanyang Technological University, Singapore. His current research interests are on social networks.

Gao Cong is an assistant professor at Nanyang Technological University, Singapore. He received his PhD degree from the National University of Singapore in 2004. Before he relocated to Singapore, he worked at Aalborg University, Microsoft Research Asia, and the University of Edinburgh. His current research interests include computer vision and machine learning.

Yifeng Zeng is a Reader at School of Computing in Teesside University. He received his PhD in 2006 from National University of Singapore, Singapore. Before he moved to Teesside University, Dr. Zeng was an assistant professor and an associate professor during 2006 - 2012 at Aalborg University, Denmark. His current research interests include computer science, particularly combinatorial design theory, coding theory, extremal set systems, and electronic design automation.
APPENDIX

A. Experiments on Weighted Networks To further confirm the performance of all comparative methods, we conduct an extra set of experiments on a weighted network of coauthor-ship (Cond)\(^2\). The weighted network represents relations between scientists posting preprints on the Condensed Matter E-Print Archive between Jan 1, 1995 and December 31, 1999. Cond has 16,264 nodes and 47,594 edges. Figure 12 repeats the performance trend of MISP and ISP methods compared to other methods (both DP and RatioDP methods will be explained in the next section) in both time constraint and conventional versions. As expected, MISP outperforms other methods in terms of a trade-off between influence spread and running times. Table 9 shows the numbers of Influence Spread in terms of a trade-off between influence spread and running times. As expected, MISP outperforms other methods in both time constraint and conventional versions. DP and RatioDP methods will be explained in the next section.

To further confirm the performance of all comparative methods, we conduct an extra set of experiments on a weighted network of coauthor-ship (Cond)\(^2\). The weighted network represents relations between scientists posting preprints on the Condensed Matter E-Print Archive between Jan 1, 1995 and December 31, 1999. Cond has 16,264 nodes and 47,594 edges. Figure 12 repeats the performance trend of MISP and ISP methods compared to other methods (both DP and RatioDP methods will be explained in the next section) in both time constraint and conventional versions. As expected, MISP outperforms other methods in terms of a trade-off between influence spread and running times. Table 9 shows the numbers of Influence Spread in terms of a trade-off between influence spread and running times. As expected, MISP outperforms other methods in both time constraint and conventional versions. DP and RatioDP methods will be explained in the next section.

Table 10–12 show the memory usage of different methods on dataset Cond. These results are consistent with those on the unweighted networks.

![Poisson Delay Distribution](image)

![Geometric Delay Distribution](image)

![Conventional Version](image)

(a) Influence Spread  
(b) Running Time

**Fig. 12.** Results in the Cond for both time constrained and conventional versions.

B. Dynamic Programming Method One alternative solution is a dynamic programming based algorithm to calculate \(\sigma_T(S)\)\(^3\). Let \(P_{i,t}\) denote the probability node \(i\) is not infected after \(t\) time-stamps, we have \(\sigma_T(S) = \sum_{i \in Y} P_{i,T}\). By denoting the probability that node \(i\) is exactly infected at time \(t\) with \(Q_{i,t}\), we have \(P_{i,t} = \sum_{j=1}^{t} Q_{i,j}\). Thus \(P_{i,t}\) can be recursively calculated as \(P_{i,t} = P_{i,t-1} \cdot Q_{i,t}\). For \(Q_{i,t}\), we can approximate it by \(Q_{i,t} \subset P_{i,t-1} \leq (1 - \prod_{j \in N_{i}(t)} \prod_{g=1}^{t} (1 - Z_{j,i,g} \leq Q_{j,t-g}))\), where \(Z_{j,i,g}\) is the probability \(j\) directly infect \(i\) with \(g\) time. By combining Algorithm 1 with the method for computing \(\sigma_T(S)\) described above, we get a solution for the Time Constrained Influence Maximization problem, which is denoted by DP in this paper. By following a similar way to extend ISP to MISP, we extend DP to RatioDP by employing a faster marginal influence spread estimation as indicated by Equation 1.

Due to the large complexity of the DP based methods, they can only solve the influence maximization problem in two small networks: Wiki and Cond networks. We show their performance in Figures 13 and 14. Figures 13(a) and 14(a) demonstrate that the ISP still achieves similar influence spread as that of the DP while the MISP outperforms the RatioDP. In Figures 13(b) and 14(b), compared to other methods, both the DP and RatioDP methods take significantly larger amount of running time on the two networks. Their consumption is even approaching that of MC, which is considered as the most time consuming algorithm in many experiments.

Tables 13 and 14 report the memory usage of all

**TABLE 9**

<table>
<thead>
<tr>
<th>Length</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cond</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**TABLE 10**

<table>
<thead>
<tr>
<th>Memory Usage in MB (Conventional Version)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
</tr>
<tr>
<td>Cond</td>
</tr>
</tbody>
</table>

**TABLE 11**

<table>
<thead>
<tr>
<th>Memory Usage in MB (T = 10, Poisson Delay Distribution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
</tr>
<tr>
<td>Cond</td>
</tr>
</tbody>
</table>

**TABLE 12**

<table>
<thead>
<tr>
<th>Memory Usage in MB (T = 10, Geometric Delay Distribution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
</tr>
<tr>
<td>Cond</td>
</tr>
</tbody>
</table>

\(^2\) http://www-personal.umich.edu/~mejn/netdata/  
\(^3\) Thanks an reviewer for suggesting the dynamic programming method
methods with $T=10$ in Wiki and Cond networks. The results show that both DP and RatioDP methods require more memory space than that needed by MISP. The extra usage is necessary since the DP methods need to backup all candidate paths in order to compute the path probabilities in a recursive way.

In summary, the DP methods are able to achieve similar influence spread in comparison to the ISP and MISP methods. However, they are deemed to be inefficient because they need to consume much more running time and memory space.

Fig. 13. Results of the DP methods in the Wiki for the time constrained version.

Fig. 14. Results in the Cond for the time constrained version.

### TABLE 13
Memory Usage in MB ($T = 10$, Poisson Delay Distribution)

<table>
<thead>
<tr>
<th>$K$</th>
<th>1</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiki</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>DC</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>MIA</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>MISP</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>ISP</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>MC</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>MIAM</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>MIAC</td>
<td>295</td>
<td>334</td>
<td>360</td>
<td>398</td>
<td>436</td>
<td>522</td>
</tr>
<tr>
<td>DP</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>RatioDP</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Cond</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>DC</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>MIA</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>MISP</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>ISP</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>MC</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>DP</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>RatioDP</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>