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

- *Bo Liu is with Facebook, Menlo Park, CA. E-mail: bol@fb.com*
- *Gao Cong and Dong Xu are with School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: {gaocong, dongxu}@ntu.edu.sg*
- *Yifeng Zeng is with School of Computing, Teesside University, UK. E-mail:Y.Zeng@tees.ac.uk*
- *Yeow Meng Chee is with School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. E-mail: {ymchee}@ntu.edu.sg*

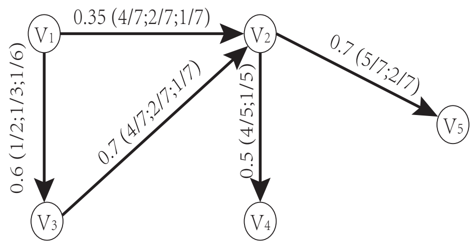


Fig. 1. An example illustrating the time constrained influence maximization.

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 history 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 unscalable 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* $\mathcal{P} = \}\cup,\mathcal{G}|$ *, a positive integer* $K < \backslash\cup\backslash$ *activating probability* $\mathcal{T}_{uv}$ / $(0,1]$ *for each* (u,v) / $\mathcal{G}$*, find a seed set* $S \rightarrow \cup$ *of* K *nodes, such that the expected number of nodes influenced by* S*,* $\sigma_T(S)$*, is maximized.*

TABLE 1
Notation Table

Notation	Definition
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$	Social Network
n	$\lvert\mathcal{V}\rvert$
m	$\lvert\mathcal{E}\rvert$
K	Number of seed nodes
S	Seed set
$N(u)$	Neighbor set of u
$\mathcal{P}_{uv}$	Probability u activates v
$\mathcal{P}_u^{lat}$	Distribution of influence propagation latency of u
$\sigma_T(S)$	Expected number of nodes influenced by S within T time units
$\sigma(S)$	Expected number of nodes influenced by S without time constraint
$\mathcal{AP}^{(t)}(u,S)$	Probability u is first activated by S at time t
$\mathcal{AP}_T(u,S)$	Probability u is activated within time T by S
$ISP(S)$	All influence spreading paths
$ISP(u,S)$	All influence spreading paths ending with u
θ	A positive influence threshold value
$ISP_{\theta,T}(S)$	Influence spreading paths with length no larger than T, probability no less than θ
$ISP_{\theta,T}(u,S)$	Influence spreading paths with length no larger than T, probability no less than θ and ending with u
$n_{\theta T}$	$max_{\lvert S\rvert \leq K}\{\lvert ISP_{\theta,T}(S)\rvert\}$

A popular model describing how influence spreads in social networks is Independent Cascade (IC) Model [18], which is widely adopted by the existing influence maximization algorithms [3]–[7], [9]. In the IC model, each node is either *active* (e.g., buying a product) or *inactive* in a social network. A node is allowed to switch from *inactive* to *active* state, but not vice versa. Given a set of seed nodes S, the IC model propagates influence in inductive steps. Let A_t be the set of nodes activated at step t, and $A_0 = S$. At step $t+1$, every node u / A_t has a single chance to activate each of its currently *inactive* neighbors v, i.e., v / $\{ {}^{t}_{i=0} A_i$. The probability that u activates v is given by the activating probability $\mathcal{T}_{uv}$ associated with edge (u,v). The influence propagation process terminates at step t, if and only if $A_t = \mathcal{A}$ In the IC model, once a node u is activated, it either activates its currently *inactive* neighbor v in the immediate next step, or does not activate v at all.

As mentioned in Section 1, influence propagation delay exists in a real-world social network, which is not captured by the IC model. We proceed to present the Latency Aware Independent Cascade (LAIC) model, which encodes the influence propagation latency information into the IC model.

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 $\mathcal{T}_{uv}\mathcal{T}_u^{lat}(\delta_t)$, where δ_t is the influencing delay and is randomly drawn from the delay distribution $\mathcal{T}_u^{lat}$. Note that a node can be activated at most once. If a node has been influenced by multiple neighbors, it is activated at the earliest activation time while the rest activations are ignored. The influence propagation process terminates at step t, iff there is no node activated after t.

Based on the proposed LAIC model, we present the time constrained influence maximization problem in Eq. 2.

Definition 2 (Time Constrained Influence Maximization)**.** *Given a social network $\mathcal{P} = \}\cup, \mathcal{G}|$, time bound T, positive integer $K < \backslash\cup\backslash$ activating probability $\mathcal{T}_{uv}$ / $(0, 1]$ for each (u, v) / $\mathcal{G}$, and latency distribution $\mathcal{T}_u^{lat}$ for each u / $\cup$, find a seed set $S \to \cup$ of K nodes, such that the expected number of nodes influenced by S within T time, $\sigma_T(S)$, is maximized under the LAIC model.*

Analogous to the conventional influence maximization problem, the time constrained version is *NP-hard*, which is shown in Theorem 1. Due to the limited space, we remove the proof and refer readers for details in [19].

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: $\mathcal{G}$, T, K, $\mathcal{P}_{uv}$ and $\mathcal{P}_u^{lat}$
Output: S
1 initialize $S = \emptyset$
2 **for** $i \leftarrow 1$ **to** K **do**
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 $\backslash S \backslash = K$. The time complexity of Algorithm 1 is $O(Kn\mathcal{V}(\sigma_T(S)))$, where n is the number of nodes in $\mathcal{P}$ and $\mathcal{V}(\sigma_T(S))$ 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 $1 \quad 1/e$, 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: $\mathcal{G}$, T, S, $\mathcal{P}_{uv}$ and $\mathcal{P}_u^{lat}$
Output: $\sigma_T(S)$
1 $v.status \leftarrow$ ***inactive***, $v.actTime \leftarrow +\infty$ for $v \in \mathcal{V} \setminus S$
2 $v.status \leftarrow$ ***active***, $v.actTime \leftarrow 0$ for $v \in S$
3 $A_0 \leftarrow S$
4 $t \leftarrow 1$
5 **do**
6   **for** $u \in A_{t-1}$ **do**
7   **for** $(u, v) \in \mathcal{E}$ *and* $v.status \neq$ ***active*** **do**
8   draw $flag$ from $Bernoulli(\mathcal{P}_{uv})$
9   **if** $flag = 1$ **then**
10   draw δ_t from $\mathcal{P}_u^{lat}$
11   **if** $v.status =$ ***inactive*** **then**
12   **if** $t + \delta_t \leq T$ **then**
13                         $v.status \leftarrow$ ***latent active***
14                         $v.actTime \leftarrow t + \delta_t$
15   **else if** $t + \delta_t < v.actTime$ **then**
16                     $v.actTime \leftarrow t + \delta_t$
17     $A_t \leftarrow \{u | u.actTime = t \cap u.status =$ ***latent active***$\}$
18     $u.status \leftarrow$ ***active*** for $u \in A_{t-1}$
19     $t \leftarrow t + 1$
20 **while** $|\{u | u.status =$ ***latent active***$\}| \neq 0$ *or* $A_t \neq \emptyset$;
21 return $\sum_{j=0}^{t} |A_j|$

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 / $N_{out}(u)$ with the probability P_{uv}. If u successfully activates v (Lines 9-20), an activating latency δ_t ($\delta_t = 0, 1, 2...$) is drawn from the discrete distribution

$\mathcal{T}_u^{lat}$ associated with node u. If v is in *inactive* state and $t+\delta_t \approx T$, v switches to *latent active* state with activating time $t+\delta_t$, which specifies when v will switch from *latent active* to *active*. If v is already in *latent active* state, v updates its activating time with the minimum of $t+\delta_t$ and its current activating time. All *latent active* nodes with activating time t automatically switch to *active* state at time step t (Line 23-24). The process terminates if and only if there are no more *latent active* 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 $\mathcal{P}$. 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 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 by incorporating influence delay information into the graph structure, based on which the definition of influence spreading path is given. Then we propose an algorithm for calculating the activation probability of a node given a seed set.

4.3.1 Augmenting Social Network with Influencing Delay Information

In the LAIC model, when a node u is first activated at time t, it tries to activate each of its outgoing neighbors v at a later time $t+\delta_t$ with a probability of $\mathcal{T}_{uv}\mathcal{T}_u^{lat}(\delta_t)$. To incorporate influence propagation delay information into the social network structure, we *logically* augment the original social network $\mathcal{P} = (\cup, \mathcal{G})$ into a directed multigraph $G_T = (V, E)$, where $V = \cup$. For each $(u,v) / \mathcal{G}$, we put T edges, $e_{uv}^1, e_{uv}^2, \times\times\times, e_{uv}^T$, from u to v in G. Each edge $e_{uv}^t (/ E)$ is guarded with two values, i.e., $length(e_{uv}^t) = t$ and $prob(e_{uv}^t) = \mathcal{T}_{uv}\mathcal{T}_u^{lat}(t)$.

Figure 2 gives the multigraph augmented from the example of social network in Figure 1 under the case of

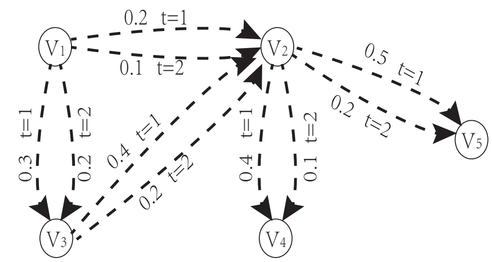


Fig. 2. The *logically* augmented multigraph with $T = 2$.

$T = 2$. 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 $\sum_{u \in V} \mathcal{ET}_T(u,S)$, where $\mathcal{ET}_T(u,S)$ is the probability that S activates u within T. It is easy to find out that $\mathcal{ET}_T(u,S) = 0$, 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 $\mathcal{ET}_T(u,S)$ 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 $p = (u_1 \overset{e_1}{\in} u_2 \overset{e_2}{\in} u_3 \ldots \overset{e_{k-1}}{\in} u_k)$ in graph G is an Influence Spreading Path, if and only if u_1 / S and $u_i \not/ S$ for $i \emptyset= 1$, where $k > 1$. For an influence spreading path p, the length of p is $\sum_{i=1}^{i=k-1} length(e_i)$, while the probability of p is $\prod_{i=1}^{i=k-1} prob(e_i)$.*

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 $\backslash ISP(u,S) \backslash$ 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 threshhold θ, 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 ($\mathcal{E}\mathcal{T}_T(u,S)$) 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 ($\mathcal{E}\mathcal{T}^{(t)}(u,S)$) (Line 3), and adds it to $\mathcal{E}\mathcal{T}_T(u,S)$ at Line 4. At Line 3, $1 \quad \mathcal{E}\mathcal{T}_T(u,S)$ is the probability u has not been activated before t, and $1 \quad \prod_{p\in ISP_{\theta,T}(u,S),length(p)=t}(1 \quad prob(p))$ is the probability u is activated at time t. At the end of each iteration t, $\mathcal{E}\mathcal{T}_T(u,S)$ is updated to store the probability u is activated before $t+1$. The loop results in the probability $\mathcal{E}\mathcal{T}_T(u,S)$ 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 $O(\backslash ISP_{\theta,T}(u,S)\backslash)$. Note that the space complexity of Function AP is also $O(\backslash ISP_{\theta,T}(u,S)\backslash)$.

Function AP

Input: $ISP_{\theta,T}(u,S)$, T
Output: $\mathcal{AP}_T(u,S)$

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

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., $ISP_{\theta,T}(u,S) \notin \mathcal{A}$ Function AP is applied to calculate the probability $\mathcal{E}\mathcal{T}_T(u,S)$ 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: $\mathcal{G}$, θ, 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 θ 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 $\quad \sigma_T(S) \leftarrow \sigma_T(S) + AP(ISP_{\theta,T}(u,S),T)$
6 return $\sigma_T(S)$

Time and space complexities Let $n_{\theta T} = max_{|S|\leq K}\}\backslash ISP_{\theta,T}(S)\backslash$, where $\backslash ISP_{\theta,T}(S)\backslash$ be the number of Influence Spreading Paths starting from S with length no less than T and probability no less than θ. 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 $\mathcal{E}\mathcal{T}_T(u,S)$ by Function AP takes $O(\backslash ISP_{\theta,T}(u,S)\backslash)$ time and $\sum_{u\in\mathcal{V}}\backslash ISP_{\theta,T}(u,S)\backslash = \backslash ISP_{\theta,T}(S)\backslash \approx n_{\theta T}$, 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 $O(Knn_{\theta T})$ 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 $u \;/\; V \quad S$. Instead of calculating $\sigma_T(S\{\ \}u|\ )$ 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., $\sigma_T(S\{\ \}v|\ ) \quad \sigma_T(S)$, 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 $\sigma_T(S\{\ \}v|\ ) \quad \sigma_T(S)$ by making a discount of $\sigma_T(\}v|\ )$ in Equation 1.

$$\sigma_T(S\{\}v|\ ) \quad \sigma_T(S) \subset \sigma_T(\}v|\ )\frac{\sum_{(v,w)\in\mathcal{E}}\mathcal{T}_{vw}(1 \quad \mathcal{T}_{Sw})\sigma_T(\}w|\ )}{\sum_{(v,w)\in\mathcal{E}}\mathcal{T}_{vw}\sigma_T(\}w|\ )} \tag{1}$$

where $\mathcal{T}_{Sw} = 1 \quad \prod_{(u,w)\in\mathcal{E},u\in S}(1 \quad \mathcal{T}_{uw})$ if $w \;/\; N(S)$; otherwise, $\mathcal{T}_{Sw} = 0$. In other words, $\mathcal{T}_{Sw}$ 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: $\mathcal{G}$, T, K, $\mathcal{P}_{uv}$, $\mathcal{P}_u^{lat}$, θ
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 $\mathcal{P}_{Sw} \leftarrow \mathcal{P}_{uw}$ for $w \in N_{out}(u)$
6 $\mathcal{P}_{Sw} \leftarrow 0$ for $w \notin N_{out}(u)$
7 **for** $k \leftarrow 1$ **to** $K-1$ **do**
8 | $u \leftarrow \arg\max_v \sigma_T(\{v\}) \frac{\sum_{(v,w)\in\mathcal{E}} \mathcal{P}_{vw}(1-\mathcal{P}_{Sw})\sigma_T(\{w\})}{\sum_{(v,w)\in\mathcal{E}} \mathcal{P}_{vw}\sigma_T(\{w\})}$
9 | $S \leftarrow S \cup \{u\}$
10 | update $\mathcal{P}_{Sw}$ 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 $O(n_{\theta T})$ time, the first *for* loop of Algorithm 4 takes $O(nn_{\theta T})$ 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 $O((K \quad 1)ne_{max})$ time, and the total running time of Algorithm 4 is $O(n(n_{\theta T} + (K \quad 1)e_{max}))$. 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 $O(Knn_{\theta T})$, 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 $O(n + m + n_{\theta T})$.

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 $v\ /\ \cup$ 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 $v\ /\ \cup$. 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.

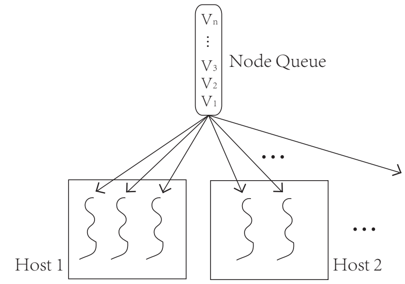


Fig. 3. The Parallelization Architecture.

As depicted in Figure 3, all nodes $\cup$ are put into the Queue, every thread repeatedly fetches 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 $\mathcal{ET}_T(u,S)$ and Algorithm 1,3,4. In what follows we brief the modifications.

Function $\mathcal{ET}_T(u,S)$ is modified into $\mathcal{ET}(u,S)$, 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 $\mathcal{ET}(u,S)$ by iterating over all $p\ /\ ISP_\theta(u,S)$ as described in Lines 2-4 of Function AP. $\mathcal{ET}(u,S)$ 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. $(1 \quad \mathcal{ET}(u,S))prob(p)$ 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 $ISP_\theta(u,S)$ is checked exactly once. Thus the time complexity of Function AP is $O(\backslash ISP_\theta(u,S)\backslash)$. The space complexity of Function AP is also $O(\backslash ISP_\theta(u,S)\backslash)$.

Algorithm 1 is modified into Algorithm 5, where $\sigma(S)$ is computed by Algorithm 6.

Function AP (For Conventional Influence Maximization Problem)

Input: $ISP_\theta(u,S)$
Output: $\mathcal{AP}(u,S)$
$\mathcal{AP}(u,S) \leftarrow 0$
for $p \in ISP_\theta(u,S)$ **do**
  $\mathcal{AP}(u,S) \leftarrow \mathcal{AP}(u,S) + (1-\mathcal{AP}(u,S))prob(p)$
return $\mathcal{AP}(u,S)$

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

Input: $\mathcal{G}$, K and $\mathcal{P}_{uv}$
Output: S
initialize $S=\emptyset$
for $i \leftarrow 1$ **to** K **do**
  $u \leftarrow \arg\max_v \sigma(S \cup \{v\}) - \sigma(S)$
  $S \leftarrow S \cup \{u\}$
return S

Algorithm 6 is modified from Algorithm 3 by filtering Influence Spreading Path by θ only at Line 2, and calling the modified $\mathcal{ET}(u,S)$ at Line 5.

Time and space complexities Let $n_\theta = max_{|S|\leq K}\}\backslash ISP_\theta(S)\backslash|$, where $\backslash ISP_\theta(S)\backslash$ is the number of Influence Spreading Paths starting from S with probability no less than θ. Following a similar analysis of Algorithm 3, we find that the time complexity of Algorithm 6 is $O(n_\theta)$. Thus the combination of Algorithm 5 and 6 takes $O(Knn_\theta)$ time. It is evident that the space complexity of Algorithm 6 is $O(n+m+n_\theta)$, where $n+m$ comes from the input social graph, and n_θ is for storing $ISP_\theta(S)$.

Algorithm 6: $\sigma(S)$ based on Influence Spreading Path (For Conventional Influence Maximization Problem)

Input: $\mathcal{G}$, θ and S
Output: $\sigma(S)$
$\sigma(S) \leftarrow 0$
get all Influence Spreading Paths with probability no less than θ by DFS.
divide them into different $ISP_\theta(u,S)$.
for *every u with non-empty $ISP_\theta(u,S)$* **do**
  $\sigma(S) \leftarrow \sigma(S) + AP(ISP_\theta(u,S))$
return $\sigma(S)$

To modify Algorithm 4 to cope with the conventional influence maximization problem, we replace all $\sigma_T(S)$ with $\sigma(S)$, 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 users 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 2
Statistics of Four Social Networks

Networks	Wiki	Epinions	Slashdot	LiveJournal
Node Number	$7,115$	$75K$	$82K$	$4.8M$
Edge Number	$103K$	$508K$	$948K$	$68.9M$
Clustering Coefficient	0.2089	0.2283	0.0617	0.3123

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 $\sigma_T(S\{\ \}v|\ )$ and $\sigma_T(S)$ 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 $\sigma_T(S\{\ \}v|\ )$ and $\sigma_T(S)$ 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 $\sigma_T(u)$ 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

1. http://snap.stanford.edu/data

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.

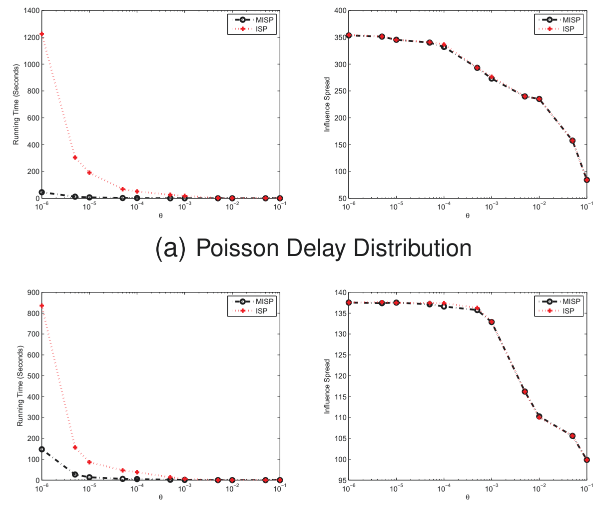
Parameter Setting The activating probability $\mathcal{T}_{uv}$ of each edge (u, v) is set by the "Weighted Cascade" policy, which is widely adopted by the existing conventional influence maximization techniques [3], [5], [6]. With "Weighted Cascade" policy, $\mathcal{T}_{uv}$ is set to be $\frac{1}{N_{in}(v)}$, 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 ($\mathcal{T}_u^{lat}$), namely *Poisson Delay Distribution* or *Geometric Delay Distribution*. For each node u / $\cup$, 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 $\frac{1}{320}$ suggested by [6], [17]. We ran them with other threshold values, which resulted in less influence spread.

Parameter θ controls the number of Influence Spreading Paths for MISP and ISP. Intuitively, a smaller value of θ 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 θ incurs a larger amount of running time. Thus there exists a tradeoff between influence spread and running time, which is tunable by θ.

To investigate the tradeoff and select an optimal value of θ, we ran MISP and ISP with different values of θ for both conventional and time constrained problems. The running time and influence spread for different θ 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 θ 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 $\theta = 10^{-5}$ ($\theta = \frac{1}{320}$), θ is set to 10^{-5} ($\frac{1}{320}$) for time constrained (resp. conventional) problem in the rest of the experimentations.



(a) Poisson Delay Distribution

(b) Geometric Delay Distribution

Fig. 4. Running time and influence spread on Wiki for different θ ($T = 10, K = 50$, Time Constrained Version).

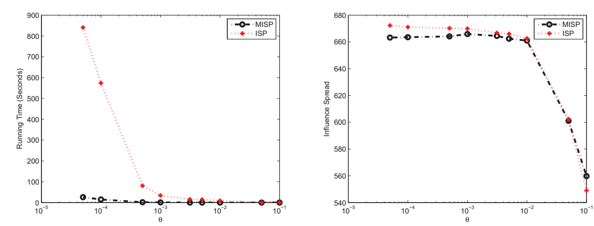
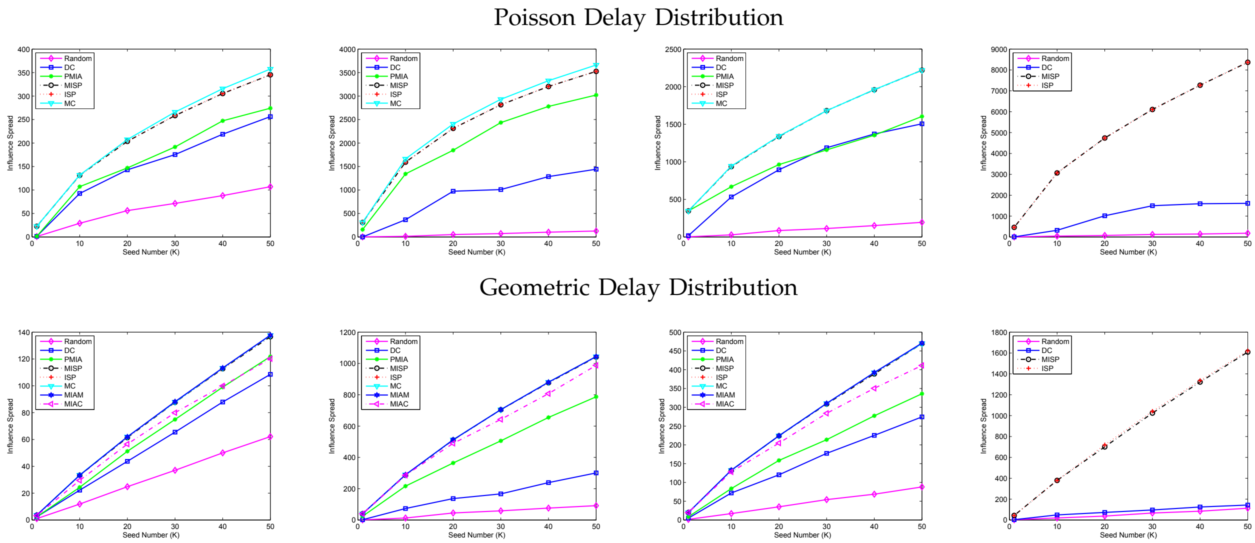


Fig. 5. Running time and influence spread on Wiki for different θ ($K = 50$, Conventional Version).

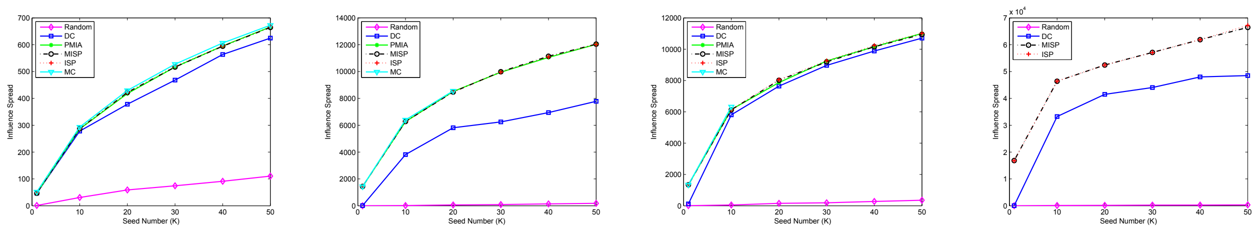
Measurement For the time constrained social influence maximization problem, a critical performance metric is the number of nodes influenced by the selected seed set within a given time. As the time constrained influence maximization problem is *NP-hard*, we are not able to get the result in polynomial 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.



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

Fig. 6. Influence spread on four real world social networks for different values of K ($T = 10$, Time Constrained Version).



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

Fig. 7. Influence spread on four real world social networks for different values of K (Conventional Version).

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-

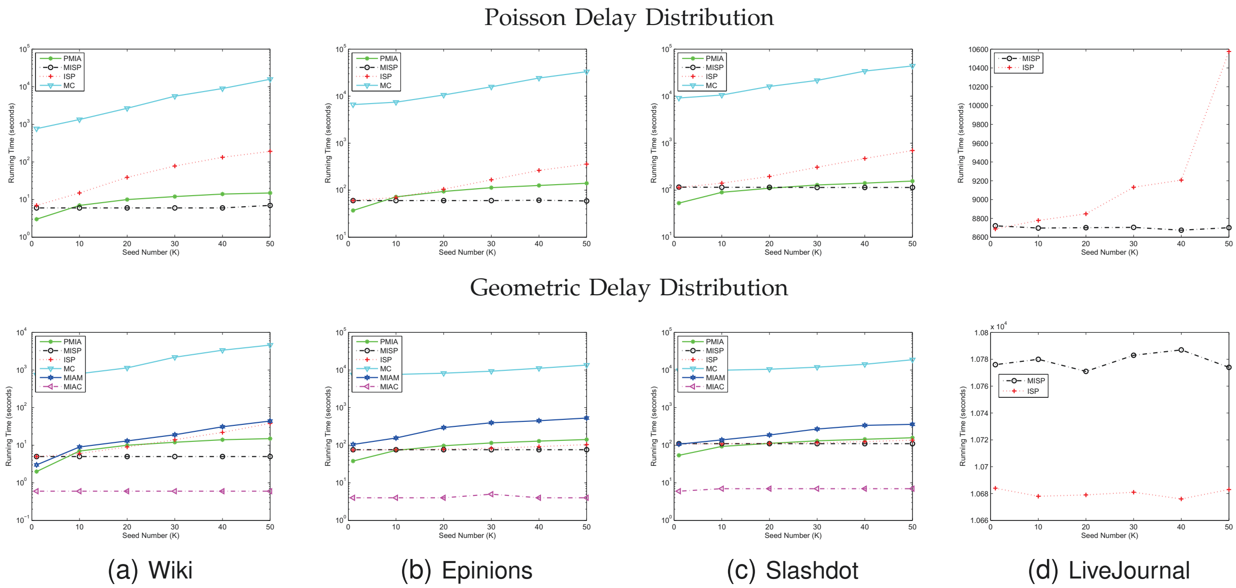


Fig. 8. Running time on four real world social networks for different values of K ($T = 10$, Time Constrained Version).

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*

Time constrained influence maximization problem Figure 8 shows the running time of different methods for each dataset with $T = 10$. As the running time for Random and DC is trivial, we do not include them to make the figure more distinguishable. When $K = 1$, ISP and MISP have similar running time, which is about two orders of magnitude faster than MC. The running time of ISP and MC increases as K increases, while the running time of MISP almost remains constant for different values of K. The results are mainly due to the fact that MISP follows the same way as ISP to select the first seed node (by Influence Spreading Path), but employs a faster marginal influence spread estimation mechanism to select the rest seed nodes. The time needed by MISP is dominated by selecting the first seed node, and thus the total running time of MISP is almost constant for different values of K. We note that MISP is nearly three orders of magnitude faster than MC when $K = 50$. For small values of K, PMIA runs faster than all other methods except DC and Random, which are not depicted, on the three datasets where PMIA can return results. However, for a large K ($K > 10$ for Wiki and Epinions, $K > 20$ for Slashdot), MISP is faster than PMIA. MIAM runs slower than MISP for almost all settings except $K = 1$ with Wiki. Though MIAC runs faster than MISP, it achieves lower influence spread as indicated in Figure 6, and cannot run with LiveJournal due to a huge memory consumption.

Table 3 shows the number of the Influence spreading paths at different lengths. Together with Figure 8, we can see that the runtime on datasets with a larger number of Influence Spreading Paths is consistently larger.

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.

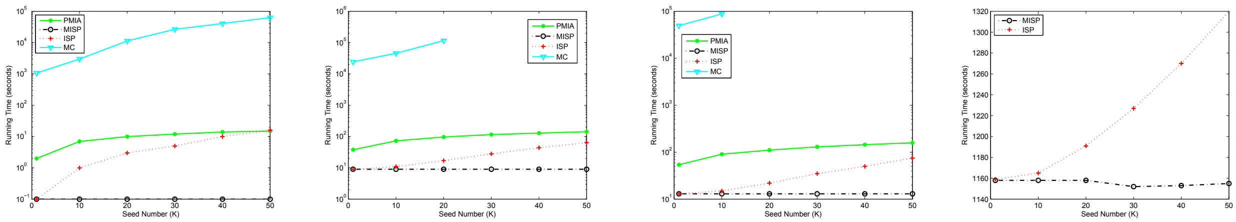
Memory Usage Tables 4, 5 and 6 show the peak memory usage of each method for different datasets with $T = 10$ for conventional and time constrained influence maximization problems, respectively. We find that Random, MISP and MC always need the same amount of memory, which is mainly occupied by the social 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 repectively. 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.

TABLE 3
Numbers of ISP Lengths over Four Datasets

Length	1	2	3	4	5	6	7	8	9	10
Wiki	36,440	55,953	90,045	122,547	146,233	155,476	148,632	135,682	118,437	97,992
Epinions	181,777	409,483	774,760	1,190,460	1,587,003	1,896,070	2,092,568	2,188,456	2,175,690	2,072,851
Slashdot	299,875	712,727	1,377,842	2,093,187	2,701,194	3,101,107	3,250,077	3,210,867	3,005,332	2,689,608
LiveJournal	24,595,582	53,644,066	100,115,927	151,621,610	198,504,194	233,224,950	251,705,564	255,912,939	247,828,529	229,682,326



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

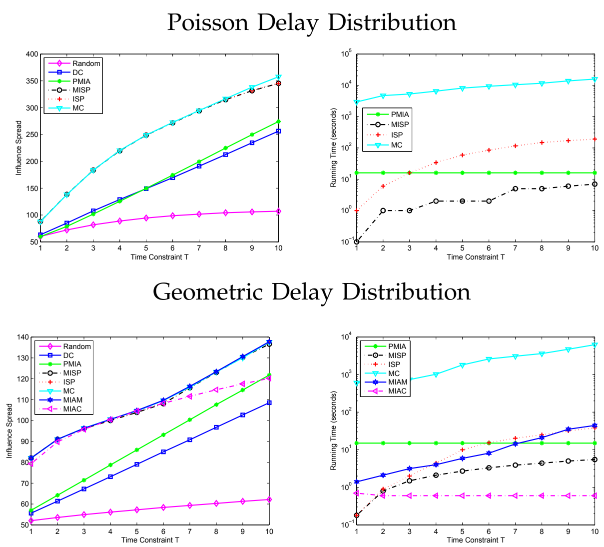
Fig. 9. Running time on four real world social networks for different values of K (Conventional Version).

TABLE 4
Memory Usage in MB (Conventional Version)

K	1	10	20	30	40	50	1	10	20	30	40	50	1	10	20	30	40	50	1	10	20	30	40	50
	Wiki						Epinions						Slashdot						LiveJournal					
Random	12	12	12	12	12	12	50	50	50	50	50	50	84	84	84	84	84	84	5785	5785	5785	5785	5785	5785
DC	15	15	15	15	15	15	74	74	74	74	74	74	119	119	119	119	119	119	8358	8358	8358	8358	8358	8358
PMIA	19	19	19	19	19	19	145	145	146	147	147	148	186	186	187	188	189	189	N.A	N.A	N.A	N.A	N.A	N.A
MISP	12	12	12	12	12	12	50	50	50	50	50	50	84	84	84	84	84	84	5785	5785	5785	5785	5785	5785
ISP	12	12	12	12	12	12	50	50	50	50	50	50	84	84	84	84	84	84	5785	5785	5785	5785	5785	5785
MC	12	12	12	12	12	12	50	50	50	N.A	N.A	N.A	84	84	N.A	N.A	N.A	N.A	N.A	N.A	N.A	N.A	N.A	N.A

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

K	1	10	20	30	40	50	1	10	20	30	40	50	1	10	20	30	40	50	1	10	20	30	40	50
	Wiki						Epinions						Slashdot						LiveJournal					
Random	13	13	13	13	13	13	51	51	51	51	51	51	85	85	85	85	85	85	5785	5785	5785	5785	5785	5785
DC	19	19	19	19	19	19	74	74	74	74	74	74	119	119	119	119	119	119	8358	8358	8358	8358	8358	8358
PMIA	19	19	19	20	20	20	145	146	147	147	148	149	186	187	188	188	189	190	N.A	N.A	N.A	N.A	N.A	N.A
MISP	13	13	13	13	13	13	51	51	51	51	51	51	85	85	85	85	85	85	5785	5785	5785	5785	5785	5785
ISP	13	19	20	21	21	21	51	57	74	78	82	83	85	99	106	138	142	147	5785	5785	5785	5785	5785	5785
MC	13	13	13	13	13	13	51	51	51	51	51	51	85	85	85	85	85	85	N.A	N.A	N.A	N.A	N.A	N.A



(a) Influence Spread (b) Running Time

Fig. 10. Results on Wiki with different T ($K = 50$).

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 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 6
Memory Usage in MB ($T = 10$, Geometric Delay Distribution)

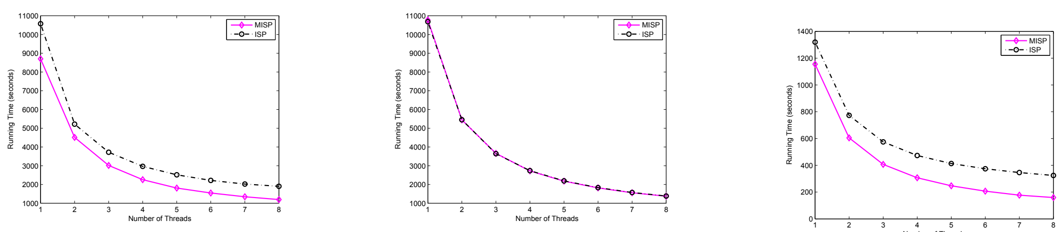
K	1	10	20	30	40	50	1	10	20	30	40	50	1	10	20	30	40	50	1	10	20	30	40	50
	Wiki						**Epinions**						**Slashdot**						**LiveJournal**					
Random	12	12	12	12	12	12	50	50	50	50	50	50	84	84	84	84	84	84	5787	5787	5787	5787	5787	5787
DC	15	15	15	15	15	15	74	74	74	74	74	74	119	119	119	119	119	119	8360	8360	8360	8360	8360	8360
PMIA	19	19	19	19	19	19	145	145	146	147	147	148	186	186	187	188	189	189	N.A	N.A	N.A	N.A	N.A	N.A
MISP	12	12	12	12	12	12	50	50	50	50	50	50	84	84	84	84	84	84	5787	5787	5787	5787	5787	5787
ISP	12	12	12	12	12	13	50	50	50	50	50	56	84	84	84	84	84	84	5787	5787	5787	5787	5787	5787
MC	12	12	12	12	12	12	50	50	50	50	50	50	84	84	84	84	84	86	N.A	N.A	N.A	N.A	N.A	N.A
MIAC	21	21	21	21	21	21	106	106	106	106	106	106	144	144	144	144	144	144	N.A	N.A	N.A	N.A	N.A	N.A
MIAM	295	334	360	398	456	521	934	1224	2004	2524	2768	3157	1031	1258	1587	2133	2531	2672	N.A	N.A	N.A	N.A	N.A	N.A

TABLE 7
The Overlaps of Seed Sets Returned by MC with different T ($K = 50$, Poisson Delay Distribution)

T	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
	Wiki										**Epinions**										**Slashdot**									
1	50	43	36	34	32	30	28	24	23	22	50	44	34	28	22	18	18	18	18	17	50	42	36	28	23	19	15	13	12	11
2		50	41	39	37	35	33	29	27	26		50	40	34	28	23	22	22	21	20		50	44	36	31	27	23	20	20	16
3			50	46	44	42	39	33	31	28			50	43	37	31	30	30	29	27			50	42	37	31	25	22	22	18
4				50	48	46	42	36	34	31				50	44	38	36	36	34	32				50	45	39	33	30	30	26
5					50	47	43	37	35	32					50	44	40	39	37	35					50	43	36	33	33	29
6						50	46	40	38	35						50	45	43	41	39						50	43	40	39	35
7							50	44	42	39							50	48	46	44							50	47	46	42
8								50	48	45								50	48	46								50	48	45
9									50	46									50	48									50	46
10										50										50										50

TABLE 8
The Overlaps of Seed Sets Returned by MC with different T ($K = 50$, Geometric Delay Distribution)

T	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
	Wiki										**Epinions**										**Slashdot**									
1	50	37	36	33	25	12	7	4	1	1	50	40	32	25	22	22	19	16	16	15	50	40	31	26	23	18	15	14	13	12
2		50	46	41	35	24	16	13	8	8		50	40	34	30	27	25	21	20	19		50	40	34	30	23	16	15	14	13
3			50	43	37	26	18	15	11	11			50	39	35	33	30	27	26	24			50	41	35	28	21	20	19	18
4				50	41	28	24	21	17	17				50	46	43	41	37	35	33				50	42	37	30	28	27	25
5					50	34	30	28	22	22					50	46	45	41	39	37					50	41	36	33	32	31
6						50	37	37	29	30						50	46	43	42	40						50	42	38	38	35
7							50	39	31	34							50	45	43	41							50	44	45	40
8								50	39	39								50	47	46								50	43	40
9									50	42									50	47									50	42
10										50										50										50



(a) Poisson Delay Distribution, $T = 10$) (b) Geometric Delay Distribution, T=10) (c) Conventional Version

Fig. 11. Running time on Livejournal with different number of threads ($K = 50$).

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 MC, 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 T, we find that the set of nodes maximizing influence spread before a given time do not 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 θ 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

[1] P. Domingos and M. Richardson, "Mining the network value of customers," in KDD, pp. 57–66, 2001.
[2] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in KDD, pp. 61–70, 2002.
[3] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in KDD, pp. 137–146, 2003.
[4] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in KDD, pp. 420–429, 2007.
[5] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in KDD, pp. 199–208, 2009.
[6] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in KDD, pp. 1029–1038, 2010.
[7] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-K influential nodes in mobile social networks," in KDD, pp. 1039–1048, 2010.
[8] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, and K. Xie, "Simulated Annealing Based Influence Maximization in Social Networks," in AAAI, pp. 127–132, 2012.
[9] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "A data-based approach to social influence maximization," in PVLDB, vol. 5, pp. 73–84, 2011.
[10] F. Bass, "A new product growth model for consumer durables," in Management Science, vol. 15, pp. 215–227, 1969.
[11] V. Mahajan, E. Muller, and F. M. Bass, "New Product Diffusion Models in Marketing: A Review and Directions for Research," in The Journal of Marketing, vol. 54, no. 1, pp. 1–26, 1990.
[12] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in WSDM, pp. 241–250, 2010.
[13] M. G. Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in KDD, pp. 1019–1028, 2010.
[14] M. G. Rodriguez, D. Balduzzi, and B. Schölkopf, "Uncovering the Temporal Dynamics of Diffusion Networks," in ICML, pp. 561–568, 2011.
[15] K. Saito, M. Kimura, K. Ohara, and H. Motoda, "Selecting Information Diffusion Models over Social Networks for Behavioral Analysis," in PKDD, pp. 180–195, 2010.
[16] K. Saito, K. Ohara, Y. Yamagishi, M. Kimura, and H. Motoda, "Learning diffusion probability based on node attributes in social networks," in ISMIS, pp. 153–162, 2011.
[17] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with time-delayed diffusion process," in AAAI, pp. 1–5, 2012.
[18] J. Goldenberg, B. Libai, and E. Muller, "Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth," in Marketing Letters, pp. 211–223, 2001.
[19] B. Liu, G. Cong, D. Xu and Y. Zeng, "Time Constrained Influence Maximization in Social Networks," in ICDM, pp. 439–448, 2012.
[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," in Mathematical Programming, vol. 14, no. 1, pp. 265–294, 1978.
[21] W. Lee, J. Kim, and H. Yu, "CT-IC: Continuously Activated and Time-Restricted Independent Cascade Model for Viral Marketing," in ICDM, pp.960-965, 2012.

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 geo-textual data management and data mining.

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 in Aalborg University, Denmark. His current research interests include intelligent agents, decision making, social networks, and computer games.

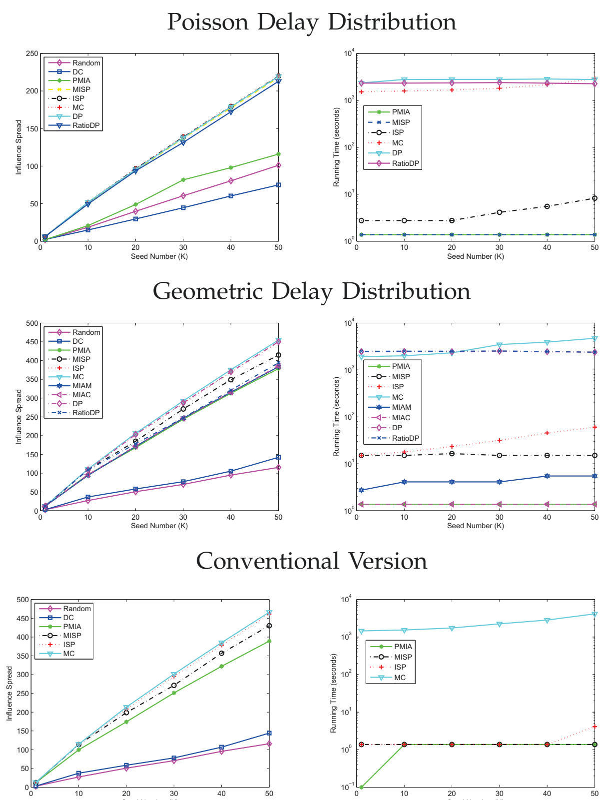
Dong Xu is currently an associate professor at Nanyang Technological University in Singapore. He received the B.Eng. and PhD degrees from University of Science and Technology of China, in 2001 and 2005, respectively. He also worked at Columbia University for one year as a postdoctoral research scientist. His research interests include computer vision and machine learning.

Yeow Meng Chee is an associate professor at Nanyang Technological University, Singapore. He received the B.Math. degree in computer science and combinatorics and optimization and the M.Math. and Ph.D. degrees in computer science, from the University of Waterloo, in 1988, 1989, and 1996, respectively. He was Program Director of Interactive Digital Media R&D in the Media Development Authority of Singapore. His research interest lies in the interplay between combinatorics and computer science/engineering, 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-ships (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 Spreading Paths with different lengths, which verifies the reasonable running time compared to that occurring in the aforementioned networks.

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



(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

2. http://www-personal.umich.edu/~mejn/netdata/
3. Thanks an reviewer for suggesting the dynamic programming method

TABLE 9
Numbers of ISP Lengths for Cond

Length	1	2	3	4	5
Cond	47,594	249,456	1,086,901	3,032,341	5,882,947
	6	7	8	9	10
	8,376,949	9,129,711	7,807,479	5,280,272	2,805,913

TABLE 10
Memory Usage in MB (Conventional Version)

K	1	10	20	30	40	50
	Cond					
Random	8	8	8	8	8	8
DC	11	11	11	11	11	11
PMIA	15	15	15	15	15	15
MISP	8	8	8	8	8	8
ISP	8	8	8	8	8	8
MC	8	8	8	8	8	8

TABLE 11
Memory Usage in MB ($T = 10$, Poisson Delay Distribution)

K	1	10	20	30	40	50
	Cond					
Random	9	9	9	9	9	9
DC	12	12	12	12	12	12
PMIA	16	16	16	16	16	16
MISP	7	7	7	7	7	7
ISP	7	8	8	9	9	9
MC	7	7	7	7	7	7

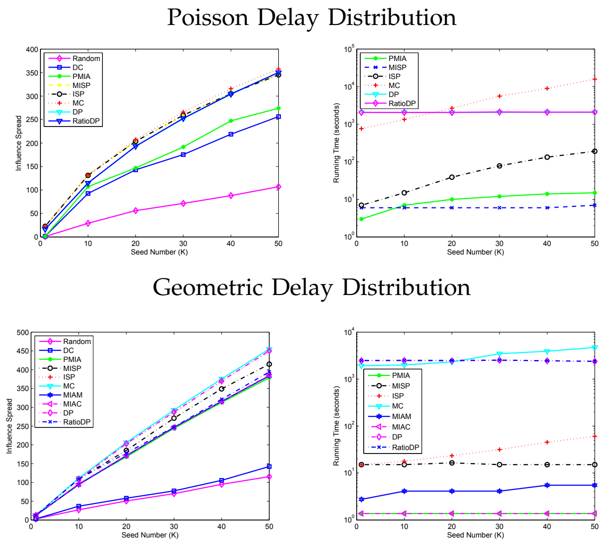
TABLE 12
Memory Usage in MB ($T = 10$, Geometric Delay Distribution)

K	1	10	20	30	40	50
	Cond					
Random	11	11	11	11	11	11
DC	14	14	14	14	14	14
PMIA	17	17	18	18	18	18
MISP	11	11	11	11	11	11
ISP	11	16	21	22	30	31
MC	11	11	11	11	11	11
MIAC	20	20	20	20	20	20
MIAM	299	305	312	317	323	330

i is not infected after t time-stamps, we have $\sigma_T(S) = \sum_{i\in\mathcal{V}} 1 \quad P_{i,T}$. By denoting the probability that node i is exactly infected at time t with $Q_{i,t}$, we have $P_{i,t} = 1 \quad \sum_{j=1}^{t} Q_{i,j}$. Thus $P_{i,t}$ can be recursively calculated as $P_{i,t} = P_{i,t-1} \quad Q_{i,t}$. For $Q_{i,t}$, we can approximate it by $Q_{i,t} \subset P_{i,t-1} \leq (1 \quad \prod_{j\in N_{in}(i)} \prod_{g=1}^{t-1}(1 \quad Z_{j,i,g} \leq Q_{j,t-g}))$. 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.

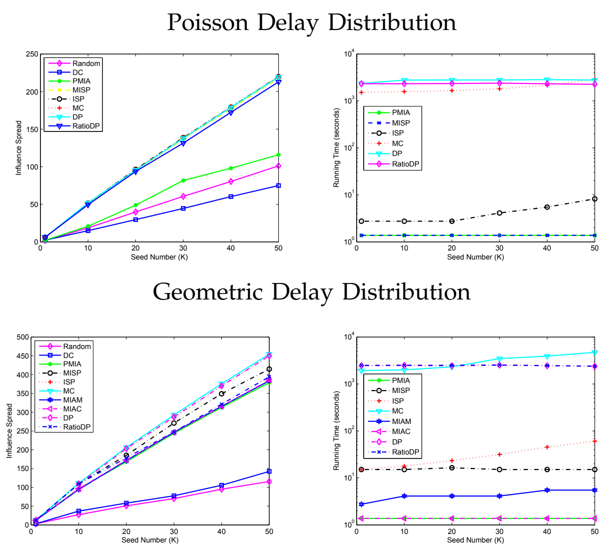
Due to the large complexity of the DP based methods, they can only solve the influence maximization problem in two small networks: Wiki and Cond networks. We show their performance in Figures 13 and 14. Figures 13(a) and 14(a) demonstrate that the ISP still achieves similar influence spread as that of 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



(a) Influence Spread (b) Running Time

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



(a) Influence Spread (b) Running Time

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

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

K	1	10	20	30	40	50
			Wiki			
Random	13	13	13	13	13	13
DC	19	19	19	19	19	19
PMIA	19	19	19	20	20	20
MISP	13	13	13	13	13	13
ISP	13	19	20	21	21	21
MC	13	13	13	13	13	13
DP	15	15	15	15	15	15
RatioDP	15	15	15	15	15	15
			Cond			
Random	9	9	9	9	9	9
DC	12	12	12	12	12	12
PMIA	16	16	16	16	16	16
MISP	7	7	7	7	7	7
ISP	7	8	8	9	9	9
MC	7	7	7	7	7	7
DP	10	10	10	10	10	10
RatioDP	10	10	10	10	10	10

TABLE 14
Memory Usage in MB ($T = 10$, Geometric Delay Distribution)

K	1	10	20	30	40	50
			Wiki			
Random	12	12	12	12	12	12
DC	15	15	15	15	15	15
PMIA	19	19	19	19	19	19
MISP	12	12	12	12	12	12
ISP	12	12	12	12	12	13
MC	12	12	12	12	12	12
MIAC	21	21	21	21	21	21
MIAM	295	334	360	398	456	521
DP	15	15	15	15	15	15
RatioDP	15	15	15	15	15	15
			Cond			
Random	11	11	11	11	11	11
DC	14	14	14	14	14	14
PMIA	17	17	18	18	18	18
MISP	11	11	11	11	11	11
ISP	11	16	21	22	30	31
MC	11	11	11	11	11	11
MIAC	20	20	20	20	20	20
MIAM	299	305	312	317	323	330
DP	14	14	14	14	14	14
RatioDP	14	14	14	14	14	14

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.