Targeted Influence Maximization in Social Networks

Chonggang Song
School of Computing
National University of Singapore
a0095629@nus.edu.sg

Wynne Hsu
School of Computing
National University of Singapore
whsu@comp.nus.edu.sg

Mong Li Lee
School of Computing
National University of Singapore
leeml@comp.nus.edu.sg

ABSTRACT

Influence maximization (IM) problem asks for a set of k nodes in a given graph G, such that it can reach the largest expected number of remaining nodes in G. Existing methods have either considered that the influence be targeted to meet certain deadline constraint, or be restricted to specific geographical region. However, if an event organizer wants to disseminate some event information on a social platform, s/he would want to select a set of users who can influence the most number of people within the neighborhood of the event location, and this influence should occur before the event takes place. Considering the location and deadline independently may lead to a less than optimal set of users. In this paper, we formalize the problem targeted influence maximization in social networks. We adopt a login model where each user is associated with a login probability and he can be influenced by his neighbors only when he is online. We develop a sampling based algorithm that returns a $(1-1/e-\varepsilon)$-approximate solution, as well as an efficient heuristic algorithm that focuses on nodes close to the target location. Experiments on real-world social network datasets demonstrate the effectiveness and efficiency of our proposed method.

1. INTRODUCTION

Influence maximization (IM) [8, 14] is a fundamental data mining problem that finds k nodes in a given network G whose adoptions of some ideas, opinions or innovations can trigger the largest expected number of adoptions by the other nodes. This problem has been extensively studied and applied to web applications such as viral marketing. Kemp et al. [14] proved that the basic influence maximation problem is NP-hard and provided a $(1 - 1/e - \varepsilon)$-approximate solution by greedily selecting k nodes where each node maximizes the marginal gain of influence spread. Since then, many methods have been developed to improve the efficiency while maintaining the quality of returned nodes [2, 5, 9, 15, 20, 21, 23].

The works in [4, 7, 18] observed that in real life, most influence maximization applications have associated deadlines. For example, a department store plans to have a Christmas sale from 21st Dec to 25th Dec. If the news of this event reaches a user after 25th Dec, then this information has zero value to the user since the sale is already over. As such, the authors in [4, 7, 18] take into consideration the deadlines, as well as possible delay in the information diffusion process, and propose heuristic algorithms to solve this deadline-aware influence maximization problem.

More recently, [16] showed that location is also important in influence maximization tasks and aimed to find k users who can maximize the number of influenced users within a specific geographical region. They extend the heuristic MIA algorithm introduced in [5] by counting only the users in the target region when computing the influence spread of each node. [25] defined a location-based influence maximization model to take into account the probability that online influence will translate to sales in a physical shop. This requires altering the target of the influence maximization to find users whose location preferences are close to the physical shop.

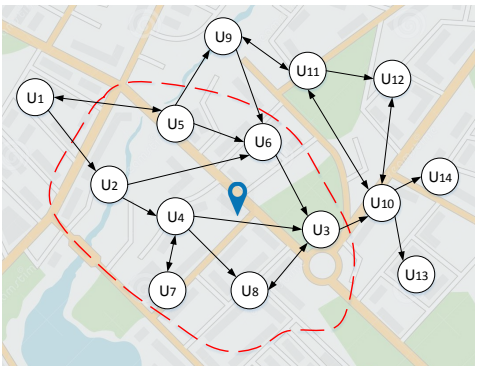


Figure 1: Example Location-Based Social Network

Figure 1 shows the locations of users on a social network where the edges denote the direction of influence. Suppose an event organizer is hosting an upcoming event at a location indicated by the blue pin, and he has the budget to broadcast the event information to only one user in the social network. To simplify discussion, we assume that if two users u and v are connected by an edge, then it takes one time point for u to influence v. For example, it will take two time points for user u_1 to influence u_4 (via u_2). Further, only users in the red circled region are willing to travel to the event location. Then location-based influence maximization methods will select u_1 as he can reach the largest number of users in the red circled region. However, if we impose a deadline of 2 time points, then u_1 can only influence 4 nodes in the region. On the other hand, time-critical influence maximization solutions will select u_3 as he can influence 7 nodes in 2 time points. Note that only 2 nodes (u_3 and u_8) are in the red circled region. Both approaches have missed the optimal user u_2 who is able to influence 5 other nodes in the targeted region within 2 time points.

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983724

This example shows the need to consider both time constraint and geographical region when solving a targeted influence maximization problem. We consider a user is *influenced* if his neighbors have propagated information of an event to him. A user is *registered* if he is *influenced* and has decided to participate in the event. Unlike traditional IM problem trying to *influence* the largest number of users online, the targeted influence maximization problem aims to identify k users that result in most *registered* participants.

In this paper, we formalize the targeted influence maximization problem which is NP-hard. We introduce the *weighted reverse reachable* (WRR) trees and develop a sampling-based approximate algorithm called Target-IM that considers the event location and deadline. This algorithm generates a pool of WRR trees and greedily selects k nodes that can cover the largest number of WRR trees. Given a social network $G = (V, E)$ where V is the set of nodes and E is the set of edges, our proposed algorithm Target-IM is able to return a $(1 - 1/e - \varepsilon)$-approximate solution in $O(k^2(|V| + |E|)\log|V|/\varepsilon^2)$ time. We further design an efficient heuristic algorithm Target-IM$^+$ by focusing on nodes close to the event location. The results of experiments on real-world datasets demonstrate that the proposed methods outperform the state-of-the-art approaches.

The rest of the paper is organized as follows. We review related works in Section 2 and define our problem in Section 3. Section 4 describes the proposed algorithms. The experimental results are presented in Section 5 and we conclude in Section 6.

2. RELATED WORK

Influence maximization problem has attracted much attention since it was introduced by Domingos et al. [8] for viral marketing. The essential idea is that by targeting a small set of users, it is possible to trigger a large range of diffusion through the word-of-mouth effect in social networks.

Existing works mostly adopt the *Independent Cascade* (IC) model and *Linear Threshold* (LT) model for information diffusion. Kempe et al. [14] formulated influence maximization as an NP-hard optimization problem and proposed a greedy algorithm that returns a $(1 - 1/e)$-approximate solution. The algorithm estimates the expected influence of each node using monte-carlo simulations, and selects the node that maximizes the marginal gain. Despite the simplicity of the greedy algorithm, it is hard to scale to large networks.

Numerous methods have been proposed to improve the efficiency of influence maximization algorithms. The works in [12, 15] exploited a lazy-forward technique to avoid the recomputation of marginal gains for all nodes at each iteration. The works in [5, 13, 19] developed heuristic techniques to speed up the computation, but these methods cannot guarantee the performance bound given in [14]. The works in [6, 20] have developed methods for scaling up the influence maximization algorithms while maintaining an approximation bound. Other works in [1, 3, 17] target users that are relevant to a certain topic or keyword to influence. However, they do not consider deadline or location information.

Recently, researchers realized the importance of geographical location in influence maximization tasks. Li et al. [16] considered influencing the set of users within a given region and extended the heuristic method in [5] to calculate the regional incremental influence of a user. They further analyzed the upper and lower bounds of a user's incremental spread to prune out users with low influence. Zhou et al. [25] introduced a function to measure the likelihood of a user's offline adoption of a product given the locations of the user and the product. The works in [10, 11, 24] study influence propagation in event-based social networks by analyzing users behaviors for predicting user interest in the nearby events. These techniques do not take into account time information.

The works in [4, 7, 18] recognized the significance of considering deadline in influence maximization tasks. Chen et al. [4] observed that when a user adopts an idea, he needs to wait for a meeting event to happen in order to influence his neighbor, thus introducing some delay in the propagation process. They proposed a MIA-M algorithm based on the notion of *maximum influence arborescence* (MIA). Each MIA keeps track of the path with the highest influence probability between each pair of nodes. Cohen et al. [7] proposed timed influence maximization by assuming that traversing an edge takes a certain amount of time and is a special case of our proposed model. The authors in [18] assign to each edge a distribution of meeting probability, indicating how likely the pair of nodes will meet with each other over a certain time frame. They proposed an *Influence Spreading Path* (ISP) algorithm for computing the influence spread of each node.

Chen et al. further extended the Independent Cascade model to incorporate log-in events to form the IC-L model [4]. They demonstrated that estimating the additional influence spread of a node under IC-L model is complicated since they need to consider the order of different nodes' log-in time. They employed a dynamic programming approach to compute the incremental influence on a node by explicitly enumerating possible log-in ordering of its in-neighbors. This approach is called MIA-L.

None of the above works have considered both deadline and location information together to find the optimal set of users in the targeted influence maximization problem.

3. PRELIMINARIES

Our work adopts the Independent Cascade model with Login events (IC-L) [4] as our diffusion model. Given a network $G = (V, E)$ where V is a set of nodes and E is a set of edges, we have an initial seed set of nodes S at time step 0. Each node $v \in G$ has a probability $login(v)$ to be online and active at each time step. When a node $v \in G$ logs in at time point $t > 0$, its neighboring node u will have a probability $p(u, v)$ of influencing v if there is an edge from u to v, u is influenced at $t' < t$, and t is the first time v logs in after t'.

In the real world, users tend to go to an event if it is near to them rather than a remote event in a different city or country. For each user $v \in G$, v is associated with a location l_v. Given an event's location γ, we have a function $f(\gamma, l_v) \in [0, 1]$ to measure the likelihood of v participating in the event. Note that $f(\gamma, l_v) \to 1$ when l_v is close to γ.

We aim to select k users such that they can influence the largest number of *registered* participants before the deadline.

DEFINITION 1 (TARGETED INFLUENCE MAXIMIZATION). *Let $G = (V, E)$ be a network where V is the set of nodes and E is the set of edges. Each node $v \in V$ is associated with a location l_v and a login probability $login(v)$ to be online, while each edge $\langle u, v\rangle$ has an influence probability $p(u, v)$. Given a deadline α and an event location γ, we aim to find a set of k users S that maximizes the expected number of registered users:*

$$\Phi(S) = \sum_{v \in I(S,\alpha)} f(\gamma, l_v)$$

where $I(S, \alpha)$ is the set of influenced users within α time points under the IC-L model.

Note that the location-based influence maximization problem in [16] and the time-critical influence maximization problem in [4] are special cases of the targeted influence maximization problem. For the former, we set $\alpha = \infty$, and for the latter, we have $f(\gamma, l_v) = 1$ for all $v \in V$.

4. PROPOSED METHOD

In this section, we present a sampling-based method called Target-IM to solve the targeted influence maximization problem.

The work in [2] introduces a *Reverse Influence Sampling (RIS)* algorithm for traditional influence maximization. RIS generates a set of Reverse Reachable (RR) sets by randomly sampling nodes in the graph. It then applies a greedy selection process based on maximum coverage [22] to find k nodes that cover the largest number of RR sets. Once a node v is selected, all RR sets that contain v are considered covered and can be removed.

In order to take deadline constraint into consideration, instead of RR sets, we define a Weighted Reverse Reachable tree to model the propagation delay incurred by login events.

DEFINITION 2 (WEIGHTED REVERSE REACHABLE TREE). *Given a graph G, let g be a graph instance of G obtained by removing each edge $\langle u, v\rangle$ in G with probability $1 - p(u, v)$. Let α be the deadline. A WRR tree for a node r, denoted as T_r, is a $(\alpha+1)$-level tree such that each path $p \in T_r$ from r to a child node v corresponds to a path from v to r in the graph instance g. Each node $v \in T_r$ is associated with the probability of v influencing r within α steps, given by $reach(v \xrightarrow{\alpha} r)$.*

Note that if there are q paths between v and r in g, we will create q copies of v, denoted as v^i, $1 \leq i \leq q$, in the WRR tree and there will be q branches in T_r from r to each of the copies of v.

Target-IM (see Algorithm 1) works in two phases. In the first phase (lines 1-3), we sample θ number of weighted reverse reachable trees. In the second phase (line 4), we greedily select k nodes that cover the most number of WRR trees generated in the first phase. However in our case, the selected node from a WRR tree T_r may only have partial influence on the root r, that is to say, other nodes in T_r may continue to exert some degree of influence on r because selecting these nodes can increase the probability of r being influenced. Hence, even if we have selected a node from T_r, we cannot remove T_r from subsequent consideration to find the next node. Finally, the algorithm returns the k nodes (line 5).

Algorithm 1: Target-IM

input : 1. Social network G
2. Deadline α
3. Location γ

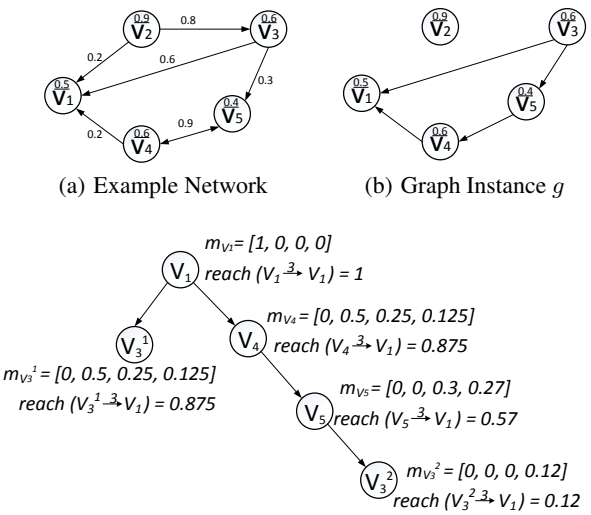
output: Seed set S

1 Initiate $\mathcal{T} = \emptyset$.
2 **while** $|\mathcal{T}| < \theta$ **do**
3 $\quad \mathcal{T} = \mathcal{T} \cup$ WRRGenerate(G, α)
4 Seed set S = GreedySelect($\mathcal{T}$)
5 Return S

In the following subsections, we elaborate on how to generate a single weighted reverse reachable tree, and describe the greedy selection process. We also give a theoretical analysis of the performance bounds of Target-IM.

4.1 Generation of WRR Tree

Given the graph G, we create a graph instance g of G by flipping a coin for each edge $< u, v >$ such that there is a probability $p(u, v)$ that the edge will be retained in g. With this graph instance, we can generate a WRR tree rooted at r as follows. We perform a breath-first traversal starting from r following the in-links. Each time we reach a node v, we create a corresponding node and add the node and its associated edge to the WRR tree. Note that if v has been visited before, a new copy of v is created.



(a) Example Network (b) Graph Instance g

(c) Generated WRR tree

Figure 2: Illustration of Generating WRR Tree.

Figure 2(a) shows an example social network where the influence probability is shown on each edge, and the login probability of each user is underlined in the node. Consider the node v_1. The probabilities of including the edges $\langle v_2, v_1\rangle$, $\langle v_3, v_1\rangle$ and $\langle v_4, v_1\rangle$ are 0.2, 0.6, and 0.2 respectively. Figure 2(b) shows a possible graph instance g obtained. Then we perform a breadth-first traversal starting from v_1. Since v_3 will be visited twice, we have two copies of v_3, namely v_3^1 and v_3^2, in the generated WRR tree rooted at v_1 (see Figure 2(c)).

Next, we describe how to compute the value associated with each node in the WRR tree, that is, $reach(v \xrightarrow{\alpha} r)$. Recall that $reach(v \xrightarrow{\alpha} r)$ denotes the probability of v influencing r within the deadline α. We use a $(\alpha + 1)$-vector m_v to aid in this computation where the j^{th} entry of m_v, denoted as $m_v[j]$, keeps track of the probability of v reaching the root in exactly j steps.

Suppose v is at the d^{th} level in the WRR tree. It will take at least d steps for v to reach r as there are at least d nodes along the path from v to r. When $j < d$, the probability that v can reach r in exactly j steps is 0.

However, when $j \geq d$, let node w be the immediate parent of v along the path from v to r, then the probability that v reaches r in exactly j steps is the probability that the parent w logs in on the $(i+1)^{th}$ step multiplied by the probability that w takes exactly $j-i-1$ steps to reach r. The probability w logs in on the $(i+1)^{th}$ step is the probability w does not log in in the first i steps and logs in on the $(i + 1)^{th}$ step, that is $(1 - login(w))^i \cdot login(w)$. The probability w takes $j-i-1$ steps to reach r is given in $m_w[j-i-1]$, hence we have

$$m_v[j] = \sum_{i=0}^{j-1}((1 - login(w))^i \cdot login(w) \cdot m_w[j - i - 1]) \quad (1)$$

where $login(w)$ is the login probability of w.

With this, we have:

$$reach(v \xrightarrow{\alpha} r) = \sum_{i=0}^{\alpha} m_v[i] \quad (2)$$

Back to our example in Figure 2. Suppose α = 3, we have $m_{v_1}[0] = 1$ since v_1 can reach itself in 0 step with a probabil-

ity 1. Consider node v_3^1 is now influenced, the probability of v_1 logging in at the next step is 0.5. Then the probability of v_3^1 influencing v_1 in 1 step is 0.5. Hence, we have

$m_{v_3^1}[0] = 0$

$m_{v_3^1}[1] = (1-0.5)^0 \times 0.5 \times m_{v_1}[0] = 0.5$

Similarly, the probability that node v_1 logs in at step 2 is $(1-0.5)^1 \times 0.5$. Then we have

$m_{v_3^1}[2] = (1-0.5)^1 \times 0.5 \times m_{v_1}[0] = 0.25$

The final WRR tree is shown in Figure 2(c).

Algorithm WRRGenerate (see Algorithm 2) gives the details. Given a network G and deadline α, we first randomly sample a node from G and create the corresponding root node r for the WRR tree (Lines 1-2). Line 3 initializes m_r and $reach(r \xrightarrow{\alpha} r)$. For each incoming edge $\langle v, w\rangle$ of w, we decide with a probability $p(v, w)$ whether it should be added to the WRR tree (Line 5). If the decision is to add the edge, all of the in-link neighbours of v are placed in a queue for subsequent processing. For the nodes that are added to the tree, we compute their $reach(v \xrightarrow{\alpha} r)$ in Lines 7-9. The algorithm terminates when all $(\alpha+1)$-level nodes have been processed.

Algorithm 2: WRRGenerate(G, α)

```
1  Randomly choose a node r uniformly and start BFS.
2  Create tree root r.
3  Initialize m_r = [1, 0, ··· , 0] and reach(r --α--> r) = 1 .
4  while breath-first traversal is within α levels of r do
5      Flip a coin with probability p(v, w) for each in-link ⟨v, w⟩.
6      if decision is YES then
7          Create node v or a copy of v if v has been visited before
8          Compute m_v[j] for each j ∈ [0, 1, ··· , α] using Equation 1.
9          Compute reach(v --α--> r) using Equation 2.
10         Place v's in-link neighbours in the processing queue.
11     else
12         Continue.
13 Return the WRR tree.
```

Note that each node in G is equally likely to be sampled by WRRGenerate. However, given a targeted location γ, a node v whose location is far away from γ should be given less consideration. As such, we should sample nodes that are close to γ as the target nodes. Further, we realize that if the expected contribution of a node in influencing the root of a WRR tree is smaller than some threshold η, then we will abandon this node and terminate the tree construction.

We design a more efficient algorithm WRRGenerate$^+$ by focusing on nodes that are closer to the target location. We first sample a node r with probability $\frac{f(\gamma,l_r)}{\sum_{v\in V} f(\gamma,l_v)}$ where nodes with higher $f(\gamma, l_r)$ value possesses a higher chance of being selected. Then we initialize m_r and start the BFS from the sampled node. For any visited node v, we create a copy of this node, compute m_v and $reach(v \xrightarrow{\alpha} r)$ as done in WRRGenerate algorithm.

Note that in Algorithm WRRGenerate, no matter how unlikely for the nodes to *register* the root, we will always perform a breadth-first traversal until α levels. However, in Algorithm WRRGenerate$^+$, we stop the traversal as soon as we realize the probability of some nodes *registering* the target node is smaller than a pre-defined threshold η. We realize that for two nodes v and w where v is an ancestor of w in T_r, it is impossible for w to have a higher probability of *registering* r than v. Hence, Algorithm WRRGenerate$^+$ will check whether $f(\gamma, l_r) \cdot reach(v^x \xrightarrow{\alpha} r) < \eta$. If so, that means all v's descendant nodes' probabilities of *registering* r cannot be larger than η. As such, we terminate the traversal and continue with the other nodes in the processing queue.

4.2 Greedy Selection

After generating a pool of WRR trees $\mathcal{T}$, the next phase is to find k nodes that cover the largest number of WRR trees. Let $T_r \in \mathcal{T}$ be a WRR tree with root node r, and $\psi(S, T_r)$ be the probability of a seed set S influencing r. For each node v in a WRR tree T_r, we maintain a value $weight(v, S, T_r)$ to indicate the contribution by v in influencing the root r where

$$weight(v, S, T_r) = \begin{cases} reach(v \xrightarrow{\alpha} r) & S = \emptyset \\ \psi((S \cup \{v\}), T_r) - \psi(S, T_r) & \text{otherwise} \end{cases} \quad (3)$$

Recall that $f(\gamma, l_r)$ gives the probability that the root node of T_r will register for an event at location γ. At each iteration, we select the node with the highest

$$\sum_{T_r \in \mathcal{T}} weight(v, S, T_r) \cdot f(\gamma, l_r)$$

to put into S. We repeat the process k times to get our seed set S.

This approach is simple but inefficient, as we need to compute the probabilities of $S \cup \{v\}$ influencing r, and S influencing r whenever we update the weight of a node v in T_r.

Careful analysis reveals that there are two cases to consider when we update the weight of a node v that does not correspond to any node in S. Let $F_S \subset S$ be the set of nodes that have no ancestors corresponding to nodes in S.

- Case 1. v is a descendant of $u \in F_S$.
  The probability of r getting influenced when both u and v are selected is the same as the probability of r getting influenced when u is selected only. This is because any influence that v can exert on r must go through u. Once u is selected, the additional contribution of v on r is 0. In other words, $weight(v, S, T_r) = 0$.

- Case 2. v is not a descendant of any node in F_S.
  Probability of r getting influenced given S
  $= 1-$ the probability that none of the nodes in S influence r
  $= 1 - \prod_{w\in F_S}(1 - reach(w \xrightarrow{\alpha} r))$

  Probability of r getting influenced given $S \cup \{v\}$
  $= 1 - \prod_{w\in F'_S}(1 - reach(w \xrightarrow{\alpha} r)))$
  where

  $$F'_S = \begin{cases} F_S \cup \{v\} - \{u\} & v \text{ is an ancestor of } u \in F_S \\ F_S \cup \{v\} & \text{Otherwise} \end{cases}$$

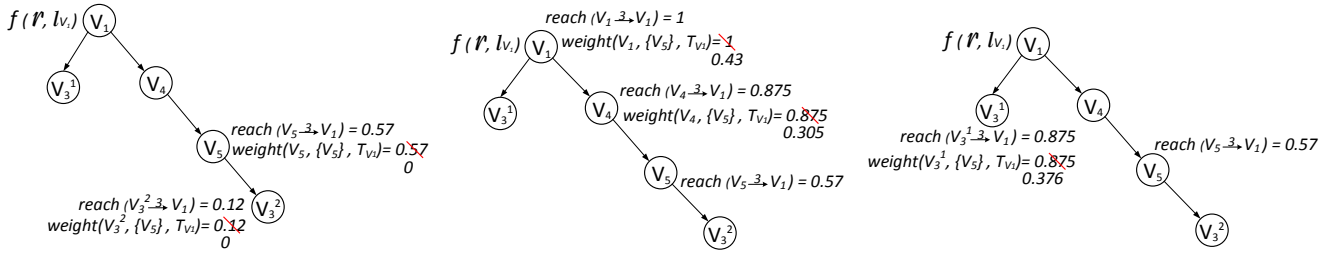
  Then we have

  $$weight(v, S, T_r) = \prod_{w\in F'_S}(1 - reach(w \xrightarrow{\alpha} r))) - \prod_{w\in F_S}(1 - reach(w \xrightarrow{\alpha} r)) \quad (4)$$

Consider the example WRR tree in Figure 3. Suppose $S = \{v_5\}$. Since v_3^2 is a descendant of v_5 (see Figure 3(a)), we set

$weight(v_3^2, \{v_5\}, T_{v_1}) = 0$.

The nodes v_1 and v_4 are ancestors of v_5 (see Figure 3(b)), their weights are updated as follows:



(a) Updating Descendants (b) Updating Ancestors (c) Updating Other Paths

Figure 3: Illustration of Updating Weights in a WRR Tree.

$$weight(v_1, \{v_5\}, T_{v_1}) = (1 - 0.57) - (1 - 1) = 0.43$$
$$weight(v_4, \{v_5\}, T_{v_1}) = (1 - 0.57) - (1 - 0.875) = 0.305$$

Finally, v_3^1 is on a different path (see Figure 3(c)) and its weight is updated as follows:

$$weight(v_3^1, \{v_5\}, T_{v_1}) = (1-0.57) - (1-0.57) \cdot (1-0.875) = 0.376.$$

Algorithm 3 gives the details. At each iteration, we select a node u with the highest $\sum_{T_r \in \mathcal{T}} weight(u, S, T_r) \cdot f(\gamma, l_r)$ value (Line 3). Then for each WRR tree T_r that involves the newly selected node u, we update the weights of the other nodes in Lines 4-15. For any node v in T_r, we first check whether the node is a copy of u. Then for each of the remaining nodes, we see whether it is a descendant of u (Lines 8-9). If not, we check whether it is an ancestor of u and update $weight(v, S, T_r)$ accordingly (Line 11-15). After updating the weights, we select the next node u' with the highest $\sum_{T_r \in \mathcal{T}} weight(u', S, T_r) \cdot f(\gamma, l_r)$ among all WRR trees. We repeat the process for k times and return the final set S as output.

Algorithm 3: GreedySelect($\mathcal{T}$, k, $f()$)

```
1  Initiate S = ∅.
2  for j = 1 to k do
3      Select u with highest probability of registering the roots
       Σ_{T_r∈T} weight(u, S, T_r) · f(γ, l_r).
4      foreach WRR tree involving u do
5          Set weight of any node copy corresponding to u to 0.
6          Identify the users in S with no ancestors in S as F_S.
7          foreach node v in the tree T_r do
8              if v is a descendant of any node in F_S then
9                  Set weight(v, S, T_r) = 0.
10             else
11                 if v is an anscestor of node u ∈ F_S then
12                     F'_S = F_S ∪ {v} − {u}.
13                 else
14                     F'_S = F_S ∪ {v}.
15                 Compute weight(v, S, T_r) with Equation 4.
16 Return S.
```

4.3 Estimation of θ

In this section, we provide an estimate of θ, the number of WRR trees to be sampled, so that the k nodes returned by Algorithm GreedySelect is guaranteed to be within $(1 - 1/e - \varepsilon)$ of the optimal solution.

Given a seed set S and a node v, Borgs et al. [2] have shown that the probability that S overlaps with a random RR set equals to the probability of S influencing v in the traditional IC model. Based on this result, we establish the following lemma.

LEMMA 1. *Let T_r be a WRR tree and S be a set of selected nodes. Suppose p is the probability that a node in S correspond to some node in T_r, and $\psi(S, T_r)$ is the probability of S influencing r. The probability that S successfully influencing r to be registered in the original graph G under IC-L model is $p \cdot \psi(S, T_r) \cdot f(\gamma, l_r)$.*

PROOF. Let R_r be the RR set generated for r on the same graph instance g by restricting the depth of BFS to α levels. According to Definition 2, the set of nodes in R_r correspond to copies of the same set of nodes in T_r. Based on the result in [2], p equals the probability of S influencing r via some path(s) in the original graph under IC model. Since the IC-L model reduces to the traditional IC model when all nodes have login probabilities 1, p is equal to the probability of S influencing r via the same path(s) under IC-L model if all nodes have login probability 1. Since T_r keeps track of these path(s) via which S influences r, so $\psi(S, T_r)$ is the probability that S influences r via these paths with different login probabilities. Taking into account the location of r in registering for an event at location γ, we have $p \cdot \psi(S, T_r) \cdot f(\gamma, l_r)$ which gives the probability of S influencing r to be registered in the original graph under IC-L model. □

Next, we define $\Psi(S, \mathcal{T})$ which gives the total expected number of root nodes in $\mathcal{T}$ that are *registered* if S is the initial set of influenced nodes as follows:

$$\Psi(S, \mathcal{T}) = \sum_{T_r \in \mathcal{T}} \psi(S, T_r) \cdot f(\gamma, l_r)$$

The function $\Psi(S, \mathcal{T})$ is monotone and submodular, indicating the greedy algorithm can produce a $(1 - 1/e)$-approximate solution [22].

THEOREM 1. *$\Psi(S, \mathcal{T})$ is monotone and submodular.*

PROOF. Clearly, given S, adding in other nodes into the S can never decrease $\psi(S, \mathcal{T})$. In addition, $f(\gamma, l_r)$ is always greater than or equal to 0. Hence, for $x \notin S$, we always have

$$\sum_{T_r \in \mathcal{T}} \psi(S \cup \{x\}, T_r) \cdot f(\gamma, l_r) \geq \sum_{T_r \in \mathcal{T}} \psi(S, T_r) \cdot f(\gamma, l_r)$$

By definition of Ψ, we have

$$\Psi(S \cup \{x\}, \mathcal{T}) \geq \Psi(S, \mathcal{T})$$

Thus Ψ is monotone.

Next, Ψ is submodular if for any two given seed sets S_1 and S_2 where $S_1 \subset S_2$, and a node $x \notin S_2$, we have $\Delta p_1 \geq \Delta p_2$ where $\Delta p_1 = \Psi(S_1 \cup \{x\}, \mathcal{T}) - \Psi(S_1, \mathcal{T})$ and $\Delta p_2 = \Psi(S_2 \cup \{x\}, \mathcal{T}) - \Psi(S_2, \mathcal{T})$.

From the definition of Ψ, we have

$$\begin{aligned}&\Psi(S_1 \cup \{x\}, \mathcal{T}) - \Psi(S_1, \mathcal{T})\\ =& \sum_{T_r \in \mathcal{T}} \psi(S_1 \cup \{x\}, T_r) \cdot f(\gamma, l_r) - \sum_{T_r \in \mathcal{T}} \psi(S_1, T_r) \cdot f(\gamma, l_r)\\ =& \sum_{T_r \in \mathcal{T}} (\psi(S_1 \cup \{x\}, T_r) - \psi(S_1, T_r)) \cdot f(\gamma, l_r)\end{aligned}$$

Similarly,

$$\begin{aligned}&\Psi(S_2 \cup \{x\}, \mathcal{T}) - \Psi(S_2, \mathcal{T})\\ =& \sum_{T_r \in \mathcal{T}} (\psi(S_2 \cup \{x\}, T_r) - \psi(S_2, T_r)) \cdot f(\gamma, l_r)\end{aligned}$$

We prove that Ψ is submodular by contradiction. Suppose $\Delta p_1 < \Delta p_2$. Then for a $T_r \in \mathcal{T}$ we have

$$\psi(S_1 \cup \{x\}, T_r) - \psi(S_1, T_r) < \psi(S_2 \cup \{x\}, T_r) - \psi(S_2, T_r)$$

that is,

$$weight(x, S_1, T_r) < weight(x, S_2, T_r)$$

Let $S1 = \emptyset$ and $S_2 = \{u\}$. Given a node $x \neq u$,

$$weight(x, S_1, T_r) = reach(x \xrightarrow{\alpha} r)$$

Further, we have
$weight(x, S_2, T_r)$

$$= \begin{cases} 0 & u \text{ is an ancestor of } x \\ reach(x \xrightarrow{\alpha} r) - reach(u \xrightarrow{\alpha} r) & u \text{ is a descendant of } x \\ (1 - reach(u \xrightarrow{\alpha} r)) \cdot reach(x \xrightarrow{\alpha} r) & \text{otherwise} \end{cases}$$

Since $weight(x, S_2, T_r)$ is always less than $reach(x \xrightarrow{\alpha} r)$, and $weight(x, S_1, T_r) = reach(x \xrightarrow{\alpha} r)$, which contradicts our assumption that $\Delta p_1 < \Delta p_2$. Thus Ψ is submodular. □

Recall that $\Phi(S)$ is defined as the number of users who register as a result of S's influence under the IC-L model. Let n be the number of nodes in the original graph G, *i.e.* $n = |V|$. We can compute the probability of S influencing a node in G to be registered as $\frac{\Phi(S)}{n}$.

Let $\mathcal{T}_S \subset \mathcal{T}$ be the subset of WRR trees that have nodes in S. Then the probability of S influencing the root of a WRR tree $T_r \in \mathcal{T}_S$ to register is given by

$$\frac{\Psi(S, \mathcal{T})}{|\mathcal{T}_S|} = \frac{\sum_{T_r \in \mathcal{T}_S} \psi(S, T_r) \cdot f(\gamma, l_r)}{|\mathcal{T}_S|}$$

Since $\frac{|\mathcal{T}_S|}{|\mathcal{T}|}$ gives the probability that S has some node corresponding to a node in a WRR tree, based on Lemma 1, we have

$$\frac{|\mathcal{T}_S|}{|\mathcal{T}|} \cdot \frac{\Psi(S, \mathcal{T})}{|\mathcal{T}_S|} = \frac{\Phi(S)}{n}$$

Hence we have

$$\frac{n}{|\mathcal{T}|} \cdot \Psi(S, \mathcal{T}) = \Phi(S) \tag{5}$$

Let M be the maximum number of registered nodes by any size-k node set in G and S^M is the set that maximizes $\Phi(S)$, *i.e.*$\Phi(S^M) = M$. Suppose S^* is the set that can *register* the largest number of tree roots in $\mathcal{T}$, that is, $S^* = \text{argmax}_S \Psi(S, \mathcal{T})$.

Based on the results established in [20], we have the following lemmas:

LEMMA 2. *If $\theta > \theta_1$ where $\theta_1 = \frac{2n \cdot log(1/\delta_1)}{M \cdot \varepsilon_1^2}$, then*

$$(n/\theta) \cdot \Psi(S^M, \mathcal{T}) \geq (1 - \varepsilon_1) \cdot M \tag{6}$$

holds with a probability of at least $(1 - \delta_1)$, $\delta_1 \in (0, 1)$, $\varepsilon_1 > 0$.

LEMMA 3. *If $\theta > \theta_2$ where $\theta_2 = \frac{(2-2/e) \cdot n \cdot log(\binom{n}{k}/\delta_2)}{M \cdot (\varepsilon - (1-1/e) \cdot \varepsilon_1)^2}$, then*

$$\frac{n}{\theta} \cdot \Psi(S, \mathcal{T}) - \Phi(S) < \varepsilon_2 \cdot M \tag{7}$$

holds with a probability of at least $(1 - \delta_2)$, $\delta_2 \in (0, 1)$, $\varepsilon_1 < \varepsilon$ and $\varepsilon_2 = \varepsilon - (1 - 1/e) \cdot \varepsilon_1$.

With the above results, we derive a bound for the approximate solution obtained by Target-IM.

THEOREM 2. *Given $\varepsilon_1 < \varepsilon$ and δ_1, δ_2 $in (0, 1)$ with $\delta_1 + \delta_2 \leq \frac{1}{n}$, we set $\theta = max\{\theta_1, \theta_2\}$ to ensure that Target-IM returns a $(1 - 1/e - \varepsilon)$-approximate solution with at least $1 - \frac{1}{n}$ probability, where θ_1 and θ_2 are determined from Lemma 2 and Lemma 3 respectively.*

PROOF. Let S be the set of nodes returned by Target-IM, $S^* = \text{argmax}_S \Psi(S, \mathcal{T})$ and $S^M = \text{argmax}_S \Phi(S)$.

According to Equation 7, we have

$$\Phi(S) > \frac{n}{\theta} \cdot \Psi(S, \mathcal{T}) - \varepsilon_2 \cdot M.$$

Since $\Psi(S, \mathcal{T})$ is a $(1 - 1/e)$-approximate solution for $\Psi(S^*, \mathcal{T})$ according to Theorem 1, we have

$$\Phi(S) > (1 - 1/e) \cdot \frac{n}{\theta} \cdot \Psi(S^*, \mathcal{T}) - \varepsilon_2 \cdot M.$$

Since S^* maximizes $\Psi(S^*, \mathcal{T})$, we know $\Psi(S^*, \mathcal{T}) \geq \Psi(S^M, \mathcal{T})$. Then we can have

$$\Phi(S) > (1 - 1/e) \cdot \frac{n}{\theta} \cdot \Psi(S^M, \mathcal{T}) - \varepsilon_2 \cdot M.$$

According to Equation 6, we know $\frac{n}{\theta} \cdot \Psi(S^M, \mathcal{T}) \geq (1 - \varepsilon_1) \cdot M$. By replacing $\frac{n}{\theta} \cdot \Psi(S^M, \mathcal{T})$ with $(1 - \varepsilon_1) \cdot M$, we have

$$\Phi(S) > (1 - 1/e) \cdot (1 - \varepsilon_1) \cdot M - \varepsilon_2 \cdot M.$$

Since $\varepsilon_2 = \varepsilon - (1 - 1/e) \cdot \varepsilon_1$ according to Lemma 3, we have

$$\begin{aligned}\Phi(S) &> (1 - 1/e) \cdot (1 - \varepsilon_1) \cdot M - (\varepsilon - (1 - 1/e) \cdot \varepsilon_1) \cdot M\\ &= ((1 - 1/e) \cdot (1 - \varepsilon_1) - \varepsilon + (1 - 1/e) \cdot \varepsilon_1) \cdot M\\ &= ((1 - 1/e) \cdot (1 - \varepsilon_1 + \varepsilon_1) - \varepsilon) \cdot M\\ &= (1 - 1/e - \varepsilon) \cdot M\end{aligned}$$

Thus we establish the bound for Target-IM. □

Theorem 2 allows us to control the result quality by setting an appropriate value for ε.

4.4 Time Complexity of Target-IM

Given an input graph $G = (V, E)$, the time complexity for generating θ WRR trees is $O(k(|V| + |E|)\log|V|/\varepsilon^2)$ based on the results in [20]. This is equivalent to the total number of edge traversed by the breath-first search. The second phase in Target-IM is to greedily select k nodes by calling Algorithm 3, GreedySelect. Since in the generation of WRR trees, each edge traversal will result in the creation of a node in the WRR tree, we have $O(k(|V| + |E|)\log|V|/\varepsilon^2)$ number of nodes in $\mathcal{T}$. In the worst case, for each iteration of Algorithm 3, we have to update the weights of all nodes in $\mathcal{T}$. This results in a time complexity of $O(k^2(|V| + |E|)\log|V|/\varepsilon^2)$. Hence the overall time complexity of Target-IM algorithm is $O(k^2(|V| + |E|)\log|V|/\varepsilon^2)$.

5. EXPERIMENTS

In this section, we present the results of experiments to evaluate the performance of the proposed methods on several real-world datasets. All the experiments are run on a linux machine with 2 Xeon E5440 2.83 GHz CPU and 16G RAM.

Datasets. We use four location-based social network datasets: GOWALLA[1], TWITTER[2], WEIBO[2] and FOURSQUARE[2]. For simplicity, we assume that the most frequent check-in location of a user u is his location l_u. Table 1 gives the total number of nodes and edges, as well as the average number of neighbours per node of these datasets.

Table 1: Characteristics of Datasets

Dataset	#Nodes	#Edges	Ave # neighbours per node
GOWALLA	26,316	106,271	8.07
TWITTER	554,372	2,402,720	8.58
WEIBO	1,020,730	16,490,916	32.3
FOURSQUARE	4,899,219	28,484,755	11.6

Methods. We compare the performance of the following methods in our experiments.

- Target-IM. This is our proposed method which computes the k nodes that maximizes the number of registered users by taking into account both deadline and user location.
- Target-IM$^+$. This method calls Algorithm WRRGenerate$^+$ instead of WRRGenerate to speed up the runtime by omitting nodes that are unlikely to participate given the event location.
- MIA-L [4]. This method computes the incremental influence spread within deadline α of a given node under the IC-L model using *maximum in-arborescence*, and does not take into consideration location information.
- Expansion [16]. This is the state-of-the-art location-aware influence maximization method. Note that this method does not consider deadline.
- IMM [20]. This is the state-of-the-art approach for the traditional influence maximization problem. IMM can be considered as a special case of Target-IM by setting $\alpha = \infty$ and $\forall v \in V$, $f(\gamma, l_v) = 1$.

Parameter Setting. We randomly generate the coordinates for the target location γ. The location function $f(\gamma, l_v)$ is set according to [16] where $f(\gamma, l_v) = 1$ if the distance between l_v and γ is within a predefined threshold, and 0 otherwise. This distance threshold is controlled by the number of nodes within a circular region centred at γ. Similar to [16], we set the number of nodes for GOWALLA to be 10^4, TWITTER and WEIBO to be 10^5, and FOURSQUARE to be 10^6.

We set the login probability of each node by randomly choosing a real number from the interval $[0, 1]$. Further, the influence probability $p(u, v)$ for each edge $\langle u, v\rangle$ is given by $1/inDegree(v)$ where $inDegree(v)$ is the number of incoming edges to node v as in [14, 21]. For Target-IM$^+$ and MIA-L, nodes influencing the target with a probability less than η will not be considered. A larger η can result in more registered users despite longer running time. Based on [4], we set $\eta = 1/320$.

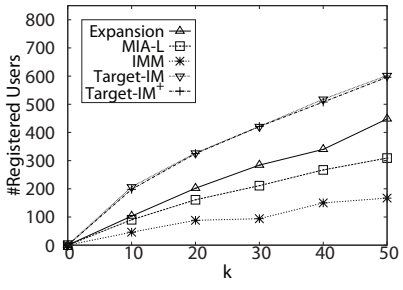
For Target-IM and Target-IM$^+$, the parameter ε controls the number of WRR trees generated. A smaller ε value indicates a tighter approximation bound but longer running time. As in [20], we fix $\varepsilon = 0.5$ in all our experiments as it provides higher efficiency.

[1]http://snap.stanford.edu/data/loc-gowalla.html
[2]http://dbgroup.cs.tsinghua.edu.cn/ligl/laim/

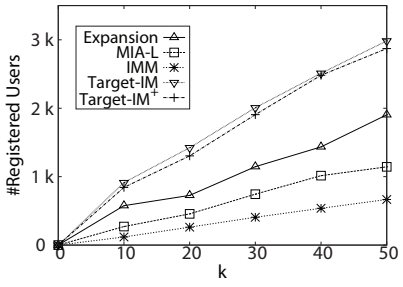
5.1 Experiments with Deadline and Location

In this set of experiments, we evaluate the performance of the various methods by running each method five times, each time with a random target location. We record the number of registered users as a result of the influence of the k users returned by these methods, and report the average results of the five runs.

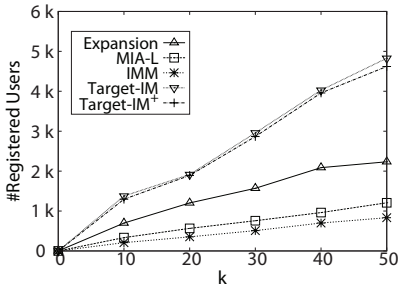
First, we set the deadline $\alpha = 10$ and vary k from 10 to 50. Figure 4 shows the results. We observe that Target-IM and Target-IM$^+$ give the best performance as they take into account both location and deadline ($+30\%$ in GOWALLA and $+80\%$ in the other three datasets). Further, the rate of increase is the steepest for Target-IM and Target-IM$^+$ as k increases.



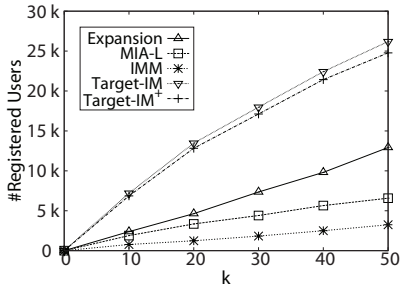
(a) GOWALLA



(b) TWITTER



(c) WEIBO



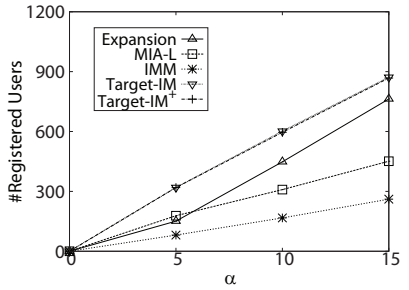
(d) FOURSQUARE

Figure 4: Effect of k ($\alpha = 10$).

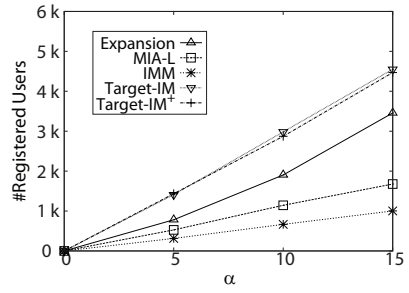
Expansion finds the set of nodes that has the highest influence within the region without considering deadline. This would include many nodes that are influenced after the deadline α, which are not the registered users in our targeted influence maximization problem. MIA-L computes the influence probability with login events and deadline. However, it does not consider the user location and may find nodes that are unlikely to be registered.

IMM has the poorest performance since it finds nodes generally with large influence spread. However, such nodes may not be the registered users when we restrict the effective nodes to a specific region and limit the propagation time.

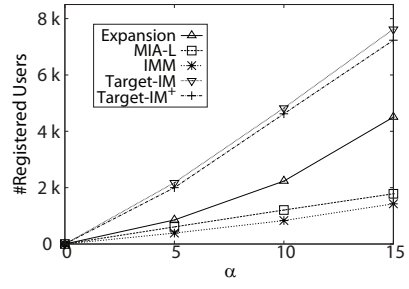
Next, we set $k = 50$ and vary the deadline α from 5 to 15. Figure 5 shows the results.



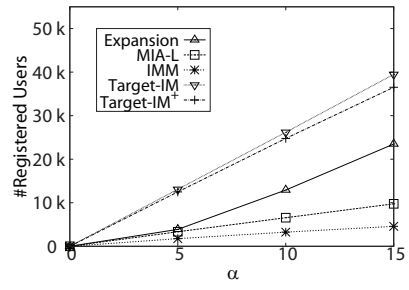
(a) GOWALLA



(b) TWITTER



(c) WEIBO



(d) FOURSQUARE

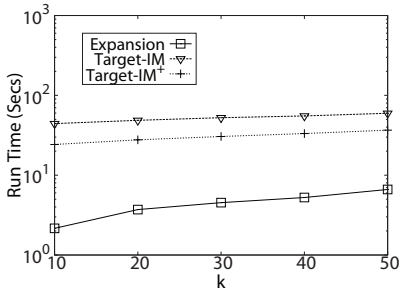
Figure 5: Effect of α ($k = 50$).

We observe that as α increases from 5 to 15, the number of registered users increases for all of the methods. However, Target-IM and Target-IM$^+$ consistently outperform the other baseline algorithms. Expansion shows the highest rate of increase in terms of the number of *registered* users when α increases. This is because Expansion finds the nodes that can *register* a large number of users if there is no deadline for the event. As α increases, the results given by Expansion slowly converges to the optimal result. When α is very small, Expansion's performance in the number of registered users is only slightly better than IMM, which considers neither location nor deadline.

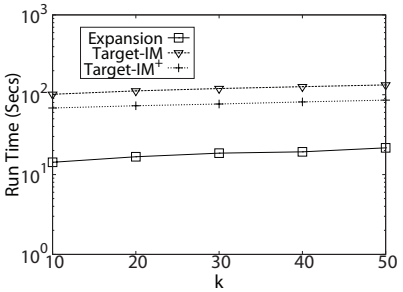
5.2 Experiments with Location only

In this set of experiments, we compare the performance of Target-IM and Target-IM$^+$ with Expansion. We set the login probability of all nodes to be 1 and deadline $\alpha = \infty$ while $f(\gamma, l_v)$ is set according to the distance between l_v and γ. In other words, the targeted influence maximization problem reduces to the location-aware influence maximization problem [16].

Figure 6 shows the runtime when we vary k from 10 to 50 on the two larger datasets WEIBO and FOURSQUARE. We observe that Expansion has the fastest execution time as it utilizes heuristics to avoid paths with low influence probability. On the other hand, Target-IM and Target-IM$^+$ consider all possible paths from u to v to estimate each node's influence more accurately, leading to a higher runtime. However, Expansion cannot guarantee any bound for the returned solution, while Target-IM has a guaranteed bound.



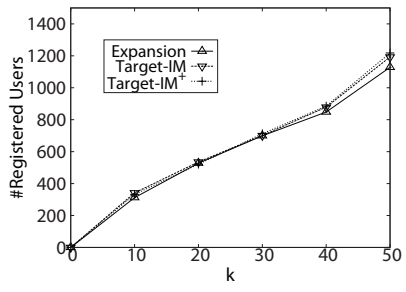
(a) WEIBO



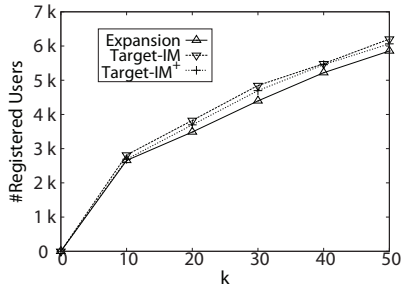
(b) FOURSQUARE

Figure 6: Runtime when $login(v) = 1$ for all v and $\alpha = \infty$.

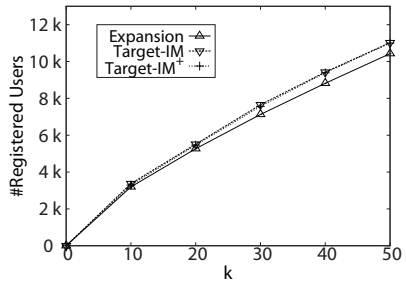
Figure 7 shows the number of registered users on all four datasets. All three methods give comparable results, although Target-IM and Target-IM$^+$ win Expansion by a small margin. Expansion considers that the best chance for v to be influenced by u through the path from u to v with the highest influence probability and ignores all other paths from u to v. As a result, Expansion may underestimate the actual influence of a node.



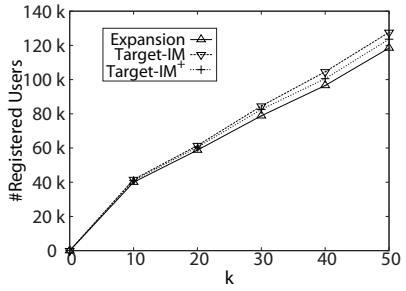
(a) GOWALLA



(b) TWITTER



(c) WEIBO



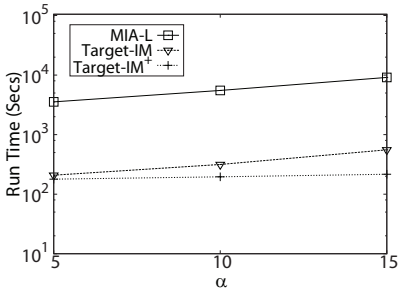
(d) FOURSQUARE

Figure 7: Performance when $login(v) = 1$ for all v and $\alpha = \infty$.

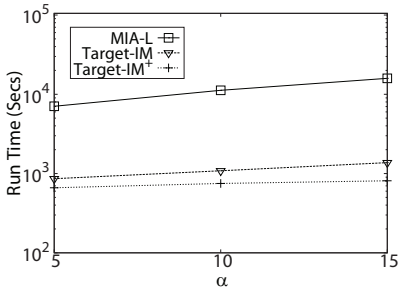
5.3 Experiments with Deadline only

Finally, we compare Target-IM and Target-IM$^+$ with MIA-L in terms of both effectiveness and efficiency without considering the deadline. We set $f(\gamma, l_v) = 1$ for all v, in other words, all of the users will register for an event regardless of its location as long as he is influenced. This setting reduces the targeted influence maximization problem to the deadline-aware influence maximization problem proposed in [4].

Figure 8 shows the runtime on the two larger datasets WEIBO and FOURSQUARE when we vary α from 5 to 15 . We see Target-IM and Target-IM$^+$ are clearly faster than MIA-L. This is because MIA-L utilizes the maximum influence in-arborecence structure to



(a) WEIBO



(b) FOURSQUARE

Figure 8: Runtime when $f(\gamma, l_v) = 1$ for all v ($k = 50$).

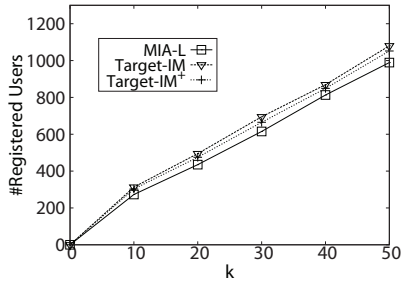
record the high influence paths. When there are multiple parent nodes pointing to the same child node in a maximum influence in-arborecence, the MIA-L algorithm needs to enumerate all possible orders of these parent nodes since different ordering of login events will result in different influence spread for each parent node. When there are many nodes that directly point to the same child node, MIA-L will incur a large amount of computation to estimate each parent node's influence on this child node.

Note that Target-IM$^+$ runs faster than Target-IM because Target-IM$^+$ avoids the computation for a node once we realize its probability of registering the root is below some threshold. When α increases, Target-IM still needs to traverse until α levels to construct each WRR tree whereas for Target-IM$^+$, we stop the breadth-first traversal as soon as we reach a node with low influence probability. Hence when α increases, Target-IM's running time increases more significantly than Target-IM$^+$'s as shown in Figure 8.

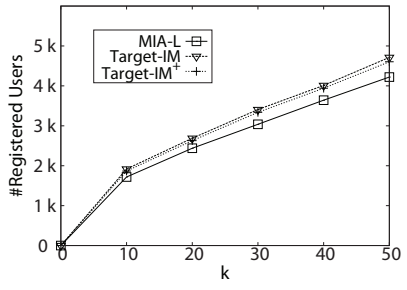
Figure 9 shows the number of registered users on all the four datasets. We observe that even though MIA-L considers deadline information, its performance is not as good as our proposed methods. This is because the path with the highest influence probability may not be the optimal path that u influences v within deadline because the nodes on the path may have low login probabilities. As a result, the set of nodes returned by MIA-L algorithm may not influence the largest number of users within deadline as compared to our proposed Target-IM and Target-IM$^+$.

6. CONCLUSIONS

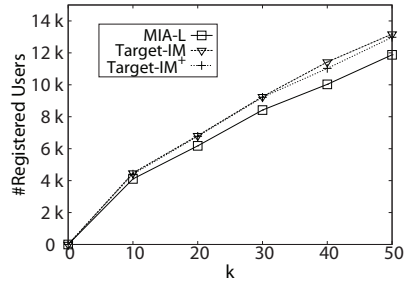
In this paper, we have examined the problem of targeted influence maximization in social networks, taking into account the temporal and geographical constraints. We introduced the notion of WRR trees and designed an algorithm called Target-IM algorithm that generates a set of WRR trees. We provided a means to estimate the number of WRR trees required in order to return a solution with $(1 - 1/e - \varepsilon)$ approximation bound. With this, we greedily selects k nodes that covers the largest number of WRR trees.



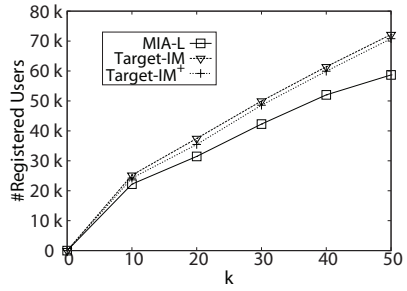
(a) GOWALLA



(b) TWITTER



(c) WEIBO



(d) FOURSQUARE

Figure 9: Performance when $f(\gamma, l_v) = 1$ for all v ($\alpha = 10$).

We further improved the WRR tree generation algorithm by omitting nodes that are unlikely to influence the target region. Extensive experimental studies have been conducted on four datasets to demonstrate the effectiveness and efficiency of Target-IM and Target-IM$^+$. Future work includes extending our problem to multi-campaign influence maximization tasks or considering event types as well as user preferences.

7. REFERENCES

[1] C. Aslay, N. Barbieri, F. Bonchi, and R. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, 2014.

[2] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, 2014.

[3] S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, and J. Tang. Online topic-aware influence maximization. *PVLDB*, 2015.

[4] W. Chen, W. Lu, and N. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. *CoRR*, 2012.

[5] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, 2010.

[6] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng. Staticgreedy: Solving the scalability-accuracy dilemma in influence maximization. In *CIKM*, 2013.

[7] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Timed influence: Computation and maximization. *CoRR*, 2014.

[8] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.

[9] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. *CoRR*, 2013.

[10] R. Du, Z. Yu, T. Mei, Z. Wang, Z. Wang, and B. Guo. Predicting activity attendance in event-based social networks: Content, context and social influence. In *UbiComp*, 2014.

[11] K. Feng, G. Cong, S. S. Bhowmick, and S. Ma. In search of influential event organizers in online social networks. In *SIGMOD*, 2014.

[12] A. Goyal, W. Lu, and L. V. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, 2011.

[13] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *ICDM*, 2012.

[14] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[15] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

[16] G. Li, S. Chen, J. Feng, K.-l. Tan, and W.-s. Li. Efficient location-aware influence maximization. In *SIGMOD*, 2014.

[17] Y. Li, D. Zhang, and K.-L. Tan. Real-time targeted influence maximization for online advertisements. *PVLDB*, 2015.

[18] B. Liu, G. Cong, D. Xu, and Y. Zeng. Time constrained influence maximization in social networks. In *ICDM*, 2012.

[19] Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. X. Yu. Influence maximization over large-scale social networks: A bounded linear approach. In *CIKM*, 2014.

[20] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, 2015.

[21] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, 2014.

[22] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.

[23] Y. Yang, X. Mao, J. Pei, and X. He. Continuous influence maximization: What discounts should we offer to social network users? In *SIGMOD*, 2016.

[24] Z. Yu, R. Du, B. Guo, H. Xu, T. Gu, Z. Wang, and D. Zhang. Who should i invite for my party?: Combining user preference and influence maximization for social events. In *UbiComp*, pages 879–883, 2015.

[25] T. Zhou, J. Cao, B. Liu, S. Xu, Z. Zhu, and J. Luo. Location-based influence maximization in social networks. In *CIKM*, 2015.