Personalized Influence Maximization on Social Networks *

Jing Guo
School of Computer Science
Beijing University of Posts and Telecommunications
Beijing, 100876, China
guojing@nelmail.iie.ac.cn

Peng Zhang
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, 100093, China
zhangpeng@iie.ac.cn

Chuan Zhou
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, 100093, China
zhouchuan@iie.ac.cn

Yanan Cao
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, 100093, China
caoyanan@iie.ac.cn

Li Guo
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, 100093, China
guoli@iie.ac.cn

ABSTRACT

In this paper, we study a new problem on social network influence maximization. The problem is defined as, given a target user w, finding the top-k most influential nodes for the user. Different from existing influence maximization works which aim to find a small subset of nodes to maximize the spread of influence over the entire network (i.e., global optima), our problem aims to find a small subset of nodes which can maximize the influence spread to a given target user (i.e., local optima). The solution is critical for personalized services on social networks, where fully understanding of each specific user is essential. Although some global influence maximization models can be narrowed down as the solution, these methods often bias to the target node itself. To this end, in this paper we present a local influence maximization solution. We first provide a random function, with low variance guarantee, to randomly simulate the objective function of local influence maximization. Then, we present efficient algorithms with approximation guarantee. For online social network applications, we also present a scalable approximate algorithm by exploring the local cascade structure of the target user. We test the proposed algorithms on several real-world social networks. Experimental results validate the performance of the proposed algorithms.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]; G.3 [**Probability and Statistics**]

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

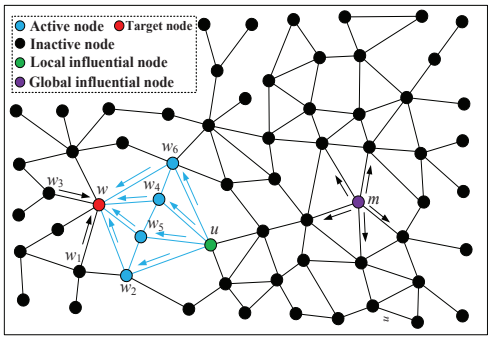


Figure 1: An Example of Local Influence

Keywords

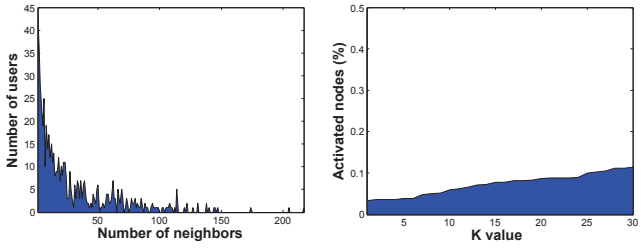
Personalization; Social Networks; Influence Maximization

1. INTRODUCTION

Social networks, as a popular and effective medium for information dissemination, play an increasingly important role in daily communication among individuals, groups and communities. Influence maximization is a problem of finding a subset of nodes in a social network that can maximize the spread of influence [1]. This research has been found useful in market recommendations, such as products, services, and innovative ideas, etc [2–5], through the powerful word-of-mouth effect in social networks.

There are a line of research work in influence maximization in social networks [1, 3, 6–26]. However, they all focus on finding global influential nodes over the entire social networks, through which the influence spread can be maximized. These work cannot answer the following question: given a specific target node w, which nodes are the most influential ones to w? Or more straightforward, who influence me most on a social network?

The above concern is critical for personalized services on social networks, such as personalized recommendation and search, target advertising, personal product promotion, personal misinformation limitation, to name a few. In these



(a) (b)

Figure 2: (a)User-neighbor Distribution (b)Top-K Influential Users and Their Influence Sphere

applications, it is essential to fully understand social influence on each specific user.

Actually, identifying top-k influential nodes for a given target user w is challenging, which cannot be easily obtained by intuitively selecting from w's neighbors, or simply choosing the global influential nodes. We use two examples for explanation.

Example 1. We take the social network in Fig. 1 for example. The objective is to identify the most influential node to a given target node w. To achieve this goal, we calculate the influence spread from each user to w [1]. From the figure, we can observe that the most influential node to w is neither its neighbors $\{w_1, \cdots, w_6\}$, nor the global influential node m, but surprisingly, its second-nearest node u !

Example 2. We use a small data set, crawled from a popular social network site in China, *weibo.sina.com*, for further explanation. The data set has 602 users and 17,595 links. We summarize the statistical characteristics of the data set in Figs. 2(a) and 2(b). From Fig. 2(a), we can observe that for a given user, there are always dozens, or even hundreds of neighbors. Thus, it is hard to determine which neighbors are the top-k influential nodes to a target user, especially when k is small. In Fig. 2(b), we summarize the influence coverage of the global top-k influential nodes (k from 1 to 30) under the independent cascade model. We can observe that even for the top-30 influential persons, there are still large blank of influence coverage, which validates our claim that finding the global influential nodes may not fulfill the personalized (local) influence maximization task.

Based on the above two observations, we summarize the challenges of the proposed problem as follows:

- Problem formulation. Each target user has her/his own local structure. How to construct the objective function by including the local structure is the first challenge.
- Algorithm. The uncertainty of influence spreading path leads to complicated measure and calculation. How to design algorithms that can balance efficiency and accuracy on large social networks is the second challenge.
- Scalability. Social networks grow fast in data volumes. How to scale to large volumes of social data is the third challenge.

[1]We will explain the details in Sections 3 and 4.

Table 1: Symbols and notations

G	Social network graph
X	Random activation result in G
Ω	Sample space of all possible activations
D	Number of simulations
U^*	Seed set
k	Number of nodes in U
δ_U	Nodes activated by U
$p_{v_1 v_2}$	Propagation probability through directed edge $<v_1, v_2>$
w	Target node
$G_{\overline{w}}$	Graph G without w
Y	Random activation result in $G_{\overline{w}}$
C_w	Neighbor set of w
$F(w, x)$	Activation result x without w
$\overline{L}$	Average number of edges
G'	Graph induced by adding a new neighbor w' to node w
m^*	Number of edges in G^*
$\mathbb{P}(X)$	Probability of X in Ω
$R_w(U)$	Local influence degree from U on w
$L(v, w)$	Corresponding local cascade community from node v to w
$1_{\{w \in X\}}$	Indicative function on the event $w \in X$

In this paper, we study the problem of local influence maximization. The objective function is to find the top-k most influential nodes for a given target user. We simulate the objective function by a random function with low variance guarantee, under the IC propagation model. As the optimization problem is NP-hard, We present a near-optimal algorithm with (1-1/e) approximation based on the submodular property of the optimization problem, and speed up the approximate algorithm's efficiency by inducing a smaller graph via the target node's neighbors. Furthermore, to scale well to large volumes of social data, we present an approximate algorithm by exploring the local cascade structure of the target user. Experimental results validate the performance of the proposed algorithms.

The rest of the paper is structured as follows. Section 2 formulates the proposed problem. Section 3 provides approximate algorithms. Section 4 presents experimental results. Section 5 surveys related work and Section 6 concludes this paper.

2. PERSONALIZED INFLUENCE MAXIMIZATION PROBLEM

In this section, we formulate the personalized influence maximization problem under the popularly used independent cascade model. Major symbols are summarized in Table 1.

2.1 Preliminaries

In this part, we introduce fundamental concepts of personalized influence calculation for a given target user w in a social network. Consider a social network $G = (V, E)$, where V and E denote nodes and edges respectively, the influence in G is propagated under the independent cascade model, IC model for short.

The IC model is the most basic model in information diffusion area. In this model, each node in G is either in active state or inactive state. The state of a node can be switched from being inactive to active, but not vice versa. When a

node v first becomes active in step t, the influence is independently propagated from node v to its currently inactive neighbors $C'_v = \{v_1, v_2, ...\}$. The node v has a single chance to activate each of its inactive neighbors. Node $v_i \in C'_v$ becomes active with the activation probability along the edge (v, v_i) at step $t+1$. No matter v succeeds in activating v_i or not, v cannot make any further attempts on v_i. As time unfolds, more and more of nodes become active, and the process runs until no more activations are possible.

Let X be a random activation result (consisting of live edges [1], through which all activated nodes can be reached from U) in the whole network G with the seed set U. Then the influence degree from set U to the target w can be measured as shown in Eq. (1),

$$R_w(U) := \mathbb{P}^U(w \in X) \tag{1}$$

where $\mathbb{P}^U$ is the probability measure via seed set U, and the notation $w \in X$ represents that w is a node in the activation result X. Hence, the influence degree $R_w(U)$ is the probability that w is successfully activated by the propagation process when the initial seed set is U.

2.2 Objective Function

The personalized influence maximization problem aims to find a seed set $U^* = \{u_1, u_2 ... u_k\} \subseteq V \backslash \{w\}$, such that $R_w(U^*) \geq R_w(U)$ for any set $U \subseteq V \backslash \{w\}$ with k nodes in the network G. Thus, the ***objective function*** of personalized influence maximization problem can be formally described as in Eq. (2),

$$U^* = \arg \max_{|U|=k, U \subseteq V \backslash \{w\}} R_w(U). \tag{2}$$

Note that the objective function of our problem is different from that of existing global influence maximization problem [1, 6] given in Eq. (3), where $R(U)$ is the influence degree from a seed set U to the nodes in the network.

$$U^* = \arg \max_{|U|=k, U \subseteq V} R(U). \tag{3}$$

The key point of our problem is how to calculate the probability $R_w(U)$, which is a $\#P-$hard problem, as we will explain in Section 2.3. An alternative method is to use Monte-Carlo simulation to evaluate $R_w(U)$. Based on Eq. (1), we have

$$R_w(U) = \mathbb{E}^U(1_{\{w \in X\}}), \tag{4}$$

where the indicative function $1_{\{w \in X\}} = 1$ if $w \in X$ stands, otherwise, $1_{\{w \in X\}} = 0$.

In particular, let Ω^U be the sample space of all possible activation results throughout the whole network G under the IC model with the seed set U, then by Eq. (4), the influence degree on w from set U can be calculated in Eq. (5),

$$R_w(U) = \sum_{x \in \Omega^U} \mathbb{P}(X = x) \cdot 1_{\{w \in x\}} \tag{5}$$

where $\mathbb{P}(X = x)$ is the probability of x in the sample space Ω^U, and $\sum_{x \in \Omega^U} \mathbb{P}(X = x) = 1$.

From Eq. (5), we can observe that $1_{\{w \in X\}}$ is an unbiased statistics to $R_w(U)$. However, its variance is somewhat large when it is used in Monte-Carlo simulation. So the next question is, how to find a better random function to simulate the objective function? To answer the question, we use Example 3 for explanation.

Example 3: We calculate the influence degree $R_w(U)$ from the seed set $U = \{①\}$ on the target node $w = ③$ in Fig. 3. The activation probability is marked along the directed edges.

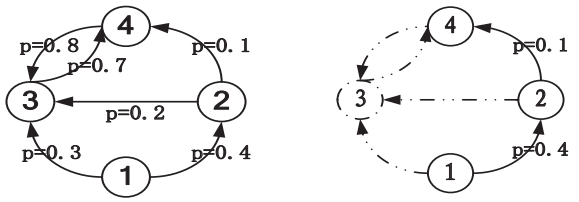


Figure 3: (a)The original graph G, (b) The generated graph $G_{\overline{w}}$ without w.

We first list the path probability in network G under seed set U, as shown in Table 2. Take $path3$ for example, the probability of $x = (① \to ② \to ④)$ in the sample space Ω^U can be calculated as follows,

$$\begin{aligned} & \mathbb{P}(X = x) \\ = \ & \mathbb{P}(X = (① \to ② \to ④)) \\ = \ & p_{1,2} \times p_{2,4} \times (1 - p_{1,3}) \times (1 - p_{2,3}) \times (1 - p_{4,3}) \\ = \ & 0.4 \times 0.1 \times (1 - 0.3) \times (1 - 0.2) \times (1 - 0.8) \\ = \ & 0.0448. \end{aligned}$$

Table 2: $R_w(U)$ under each path in G

No.	x	$\mathbb{P}(X = x)$	$F(w, x)$
1.	①	0.42	①
2.	① → ②	0.2016	① → ②
3.	① → ② → ④	0.00448	① → ② → ④
4.	① → ② → ④ → ③	0.01792	① → ② → ④
5.	① → ② → ③	0.01512	① → ②
6.	① → ② → ③ → ④	0.03528	① → ②
7.	① → ② → ④ ↓ ③	0.0056	① → ② → ④
8.	① → ③	0.054	①
9.	① → ③ → ④	0.126	①
10.	① → ② ↓ ③	0.0324	① → ②
11.	① → ② → ④ ↓ ③	0.0036	① → ② → ④
12.	① → ③ → ④ ↓ ②	0.0756	① → ②
13.	① → ② → ④ ↘ ↗ ③	0.0084	① → ② → ④
Sum.	− − −	1	− − −

From Table 2, we have two observations. First, $R_w(U)$ can be calculated as shown in Eq. (5), by summing up all the probabilities $\mathbb{P}(X = x)$ with $w \in x$, i.e.,

$$R_w(U) = \sum_{w \in x} \mathbb{P}(X = x) = \sum_{x = x_4, ... x_{13}} \mathbb{P}(X = x) = 0.37392 \tag{6}$$

Second, we remove the target node w from each activation result x in G. Let $F(w,x)$ denote the activation result without w, then by simply merging identical terms in the fourth column of Table 2, we get the probability for each merged result, including $\mathbb{P}(F(w,x) = ①) = 0.6$, $\mathbb{P}(F(w,x) = ① \rightarrow ②) = 0.36$, $\mathbb{P}(F(w,x) = ① \rightarrow ② \rightarrow ④) = 0.04$.

Figure 4: Graph without the target node

Obviously, the distribution of $F(w,x)$ is identical with the random activation result (which can be denoted as Y) throughout the whole network without the target node, as shown in Fig. 4.

For a given target node $w \in V$, each neighbor has its own propagation probability to the target. Thus, the personalized influence in network G on the user w can be calculated by combining two parts:

- Propagation information from the seed set U to the activated (persuaded) neighbors of w;
- Propagation probability from the activated (persuaded) neighbors to the target user w.

Therefore, we can obtain the result by taking Y into account under the IC model,

$$\begin{aligned} & \sum_y \mathbb{P}(Y=y)\Big(1-\prod_{v\in y}(1-p_{vw})\Big) \\ = \; & 0.6\times 0.3 + 0.36\times(1-0.7\times 0.8) \\ & +0.04\times(1-0.7\times 0.8\times 0.2) \\ = \; & 0.37392 \end{aligned} \tag{7}$$

□

We can observe that the results in Eqs. (6) and (7) are the same. This is not a coincidence, we will theoretically prove the point. Actually, based on the property of the conditional probability [27], we have

$$\begin{aligned} R_w(U) &= \mathbb{P}^U(w\in X) \\ &= \mathbb{E}^U\Big[\mathbb{P}^U\big(w\in X|F(w,X)\big)\Big] \\ &= \mathbb{E}^U\Big[1-\prod_{v\in F(w,X)}(1-p_{vw})\Big] \\ &= \mathbb{E}^U\Big[1-\prod_{v\in Y}(1-p_{vw})\Big]. \end{aligned}$$

Namely, $1-\prod_{v\in Y}(1-p_{vw})$ is an unbiased statistics of $R_w(U)$. Moreover, it has smaller variance than $1_{\{w\in X\}}$. In fact, we have

$$\begin{aligned} & \mathbb{D}^U\Big[1-\prod_{v\in Y}(1-p_{vw})\Big] \\ = \; & \mathbb{E}^U\Big[\big(1-\prod_{v\in Y}(1-p_{vw})-R_w(U)\big)^2\Big] \\ = \; & \mathbb{E}^U\Big[\big(1-\prod_{v\in Y}(1-p_{vw})\big)^2\Big]-\big[R_w(U)\big]^2 \\ = \; & \mathbb{E}^U\Big[1-2\prod_{v\in Y}(1-p_{vw})+\big(\prod_{v\in Y}(1-p_{vw})\big)^2\Big] \\ & -\big[R_w(U)\big]^2 \\ = \; & \mathbb{E}^U\Big[-\prod_{v\in Y}(1-p_{vw})+\big(\prod_{v\in Y}(1-p_{vw})\big)^2\Big] \\ & +R_w(U)-\big[R_w(U)\big]^2 \\ < \; & R_w(U)-\big[R_w(U)\big]^2 \\ = \; & \mathbb{E}^U\Big[1^2_{\{w\in X\}}\Big]-\big[R_w(U)\big]^2 \\ = \; & \mathbb{D}^U\Big[1_{\{w\in X\}}\Big]. \end{aligned}$$

To sum up, we can obtain Theorem 1 as follows,

THEOREM 1. *The statistics $1-\prod_{v\in Y}(1-p_{vw})$ is unbiased for evaluating $R_w(U)$, and it has smaller variance than $1_{\{w\in X\}}$, i.e.,*

$$\mathbb{E}^U\Big[1-\prod_{v\in Y}(1-p_{vw})\Big] = R_w(U),$$

$$\mathbb{D}^U\Big[1-\prod_{v\in Y}(1-p_{vw})\Big] < \mathbb{D}^U\Big[1_{\{w\in X\}}\Big].$$

The influence degree from U to the target w can be measured as in Eq. (8), which is demonstrated effective to measure the personalized influence degree with provable smaller variance by Theorem (1).

$$R_w(U) = \mathbb{E}^U\Big[1-\prod_{v\in Y}(1-p_{vw})\Big]. \tag{8}$$

2.3 Properties of Objective Function

By combining Eqs.(2) and (8), we obtain the objective function in Eq.(9),

$$U^* = \arg\max_{|U|=k, U\subseteq V\setminus\{w\}} \mathbb{E}^U\Big[1-\prod_{v\in Y}(1-p_{vw})\Big]. \tag{9}$$

There are two challenges, given in Theorems 2 and 3, in solving the objective function in Eq.(9).

THEOREM 2. *The problem*

$$U^* = \arg\max_{|U|=k, U\subseteq V\setminus\{w\}} R_w(U)$$

is NP-hard, and an instance of the classical set cover problem can be viewed as a special case of this problem.

PROOF. Consider a collection of subsets $S_1, S_2, ..., S_m$ of a ground set $U = \{u_1, u_2, ..., u_c\}$, the set cover problem is to find whether there exist k of the subsets, whose union is equal to the ground set U (We assume that $k<c<m$).

Given an arbitrary instance of the set cover problem, we can define a directed bipartite graph with $|V|+|C_w|$ nodes, as shown in Fig.5, where V denotes the node set in graph $G_{\overline{w}}$, $|V| = m$, and each node $i \in V$ corresponding to a set S_i; C_w denotes the neighbors of the target node w, $|C_w| = c$, and each node $j \in C_w$ corresponding to a element u_j. Each edge $< i, j >$ with propagation probability $p_{ij} = 1$, if $u_j \in Si$. In this way, the set cover problem is equivalent to deciding if

there is a k-node seed set U in this bipartite graph that can active all C_w. If any k-node seed set can reach this target, then the Set Cover problem must be solvable. □

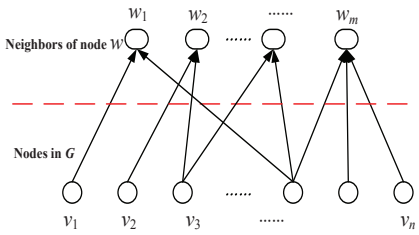


Figure 5: Bipartite graph with $|V| + |C_w|$ nodes

As the given problem is NP-hard, it is impractical to solve with brute force, especially when the network is large. To solve the problem with provable accuracy, we prove that the objective function has the sub-modular property, which can be exploited to develop $(1 - 1/e)(\approx 63\%)$ approximate algorithms (e.g., greedy algorithms).

THEOREM 3. *The computing*

$$R_w(U) = \mathbb{E}^U\Big[1 - \prod_{v \in Y}(1 - p_{vw})\Big]$$

is #P-hard, as it can be viewed as a reduction from an instance of the classical counting problem of s-t connectness in a directed graph.

PROOF. Consider the classical counting problem of s-t connectness in a directed graph. This problem is equivalent to computing the probability that s is connected to t, when we set the connection probability of each edge in G with p=1/2.

We define $R_w(s, G)$ to denote the influence degree from a given set {s} to a target node w in G, and define $R_w(s, G')$ to denote the influence degree from a given set {s} to a target node w in G', where G' is a graph induced by adding a new neighbor w' of w, and two directed edges $<w', w>$ with $p_{w',w}$=1/2 and $<t, w'>$ with $p_{t,w'}$=1 in original G. Therefore, $R_w(s, G') - R_w(s, G) = (1 - R_w(s, G)) \cdot p_{st,G} \cdot p_{tw} \cdot p_{w'w}$, where $p_{st,G}$ denotes the propagation probability from the seed set s to t in G. As a result, $R_w(s, G) - R_w(s, G')$ is related to the probability that s connects to t. If $R_w(s, G) - R_w(s, G')$ is solved, then the s-t connectness problem is solvable. □

As the local influence spread is #P-hard, we present two approximation solutions:

(I) we use *Monte-Carlo method* to simulate the influence cascade. The number of possible activation results is exponential, simulating the random diffusion process with sufficient times can obtain arbitrarily close approximations to $R_w(U)$ with high probability. Let D denote the number of simulation times, Y denote the random activation result (consisting of live edges) with seed set U in $G_{\overline{w}}$, $\Omega^U_{\overline{w}}$ denote a sample space of all possible activations. On the basis of Eq. (8), $R_w(U)$ can be computed as in Eq. (10),

$$R_w(U) = \sum_{y \in \Omega^U_{\overline{w}}} \mathbb{P}(Y = y)(1 - \prod_{v \in y}(1 - p_{vw})) \qquad (10)$$

For simplicity, we denote $1 - \prod_{v \in y}(1 - p_{vw})$ as $R_w(U, y)$. With a limited simulation number N, we get $R_w(U)$ as shown in Eq. (11).

$$R_w(U) \approx \frac{1}{D}\sum_{i=1}^{D} R_w(U, y_i) \qquad (11)$$

(II) we construct a local cascade community that consists of only *the shortest paths* between each node in the graph and the target node, and restrict computations within this community only. This method significantly reduces the uncertainty during the influence propagation, and shrinks computation cost. In doing so, the approximate estimations of the local influence propagation can be efficiently calculated.

3. APPROXIMATE ALGORITHMS

In this section, we present two near-optimal algorithms and an on-line algorithm for the personalized influence maximization.

3.1 Greedy algorithm

We first prove that the optimization problem in Eq. (2) has the *sub-modular* [1,28] property. Based on this property, we then develop a greedy hill-climbing solution, referred to as the *local greedy algorithm* (LGA for short).

THEOREM 4. *The objective function in Eq. (2) has the sub-modular property.*

PROOF. First, we show that $R_w(U, Y)$ is sub-modular. Let A, B denote the node set in graph G ($A \subseteq B \subseteq V$), $\triangledown R_w(A_v, Y) = R_w(A \cup v, Y) - R_w(A, Y)$ denote the marginal influence degree on the target node w in $R_w(v, Y)$ that are not already in the union $\bigcup_{u \in A} R_w(u, Y)$. Obviously, $\triangledown R_w(A_v, Y)$ is no less than $\triangledown R_w(B_v, X)$, which can be converted as $R_w(A \cup v, Y) - R_w(A, Y) \geq R_w(B \cup v, Y) - R_w(B, Y)$, this satisfies the definition of sub-modularity. As sub-modular functions are closed under nonnegative linear combinations, the objective function in Eq. (2) has the sub-modular property. □

For a non-negative, monotone sub-modular function r, let S be a set of size k obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value. Let S^* be a set that maximizes the value of r over all k-element sets. Then $r(S) \geq (1 - 1/e)r(S^*)$. In other words, S provides a $(1 - 1/e)$-approximation [1, 28].

$$U = \bigcup_{k} \arg\max_{v \in V \setminus U} R_w(U \cup \{v\}) - R_w(U) \qquad (12)$$

Now we introduce the LGA algorithm. The algorithm starts with an empty seed set, and repeatedly adds a node that gives the maximum marginal gain into the set, until k nodes has been obtained.

We summarize LGA in Algorithm 1. The input is a graph G, a target node w and a positive number k which is the expected number of nodes to be selected in the seed set U. We use *RanCas* to denote a random process in $MonteCarlo$ stochastic simulation.

LGA first adds one node as seed each round, such that this node can maximize the marginal influence degree on w, as

Algorithm 1: Local Greedy Algorithm (LGA)

Input : (1) k, a positive number (2) w, a target node (3) G, a graph (4) D, the number of simulation times
Output: U, the top-k influential nodes for w

$U \longleftarrow \phi$
Create $G_{\overline{w}}(G, w)$
while $|U| < k$ **do**
  for *each node* $v \in V \setminus U$ **do**
    $R_w(U \cup \{v\}) = 0$
    for $i = 1$ *to* D **do**
      RanCas $Y(i, v, G_{\overline{w}})$
      $R_w(U \cup \{v\}) += R_w(U \cup \{v\}, Y)$
    end
    $R_w(U \cup \{v\}) = R_w(U \cup \{v\})/D$
  end
  $U = U \bigcup \arg \max_{v \in V \setminus U} R_w(U \cup \{v\})$
end
Output U

well as the maximize the influence spread on w together with current seed set. For this purpose, the influence spread of each node is estimated with D repeated simulations of RanCas $Y(i, v, G_{\overline{w}})$. Due to the sub-modularity of the objective function, LGA guarantees a solution which can achieve at least a constant fraction $(1 - 1/e)$ of the optimal score.

We now discuss the complexity of LGA. In one random simulation, the influence spread calculation from each node in $G_{\overline{w}}$ takes $O(m)$ time, where m is the number edges in $G_{\overline{w}}$. LGA estimates the random diffusion process with D repeated simulations. Thus, the time complexity of selecting one seed in LGA is $O(nDm)$, where n is the number of nodes in $G_{\overline{w}}$. For k seed nodes in U , the time complexity is $O(knDm)$.

3.2 Speeding-up algorithm

We present a more *efficient local greedy algorithm* (ELGA for short), based on the above LGA algorithm. Compared to LGA, ELGA reduces the time complexity from $O(knDm)$ to $O(kD \cdot (m + m^*))$, where m^* denotes the average number of edges of all induced smaller graphs.

In fact, in one random activation result Y, whether the influence propagated through an edge or not [11] is determined, so we don't need to simulate the influence cascade for all the nodes in graph $G_{\overline{w}}$ one by one, as the LGA algorithm does. An alternative method is to induce a graph G_i^* from the original graph $G_{\overline{w}}$ in each random process i first, by simulating the influence cascade on all edges in graph $G_{\overline{w}}$, and then calculate the $R_w(U, G_i^*)$ for nodes in the graph in one time, shown as Fig.6.

The following steps also are used to further increase the efficiency of ELGA algorithm.

First, further reduce the scale of graph G_i^*. As a seed node always influences the target node w through w's neighbors, we can remove the nodes and edges that can not reach w's neighbors in G_i^*, which will have no impact on the calculation results, to further reduce the scale of graph G_i^*.

Second, reduce unnecessary calculation. If a node v is already activated by the current seed set U, then the node v

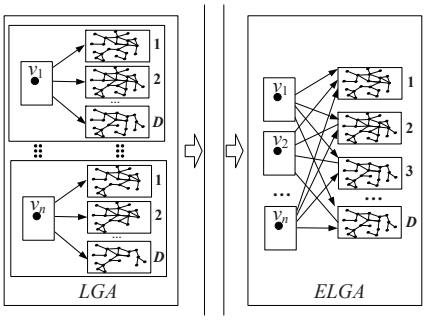


Figure 6: The Comparison between LGA and ELGA

has no contribution to our influence calculation [10] on the target w. As a result, for each node $v \in V \setminus U$,

$$R_w(U \cup \{v\}, G_i^*) = \begin{cases} R_w(U, G_i^*) & if\ v \in \delta_U \\ R_w(U, G_i^*) + R_w(v, G_i^*) & if\ v \notin \delta_U \end{cases} \quad (13)$$

where δ_U denotes the nodes activated (reachable) by the current seed set U.

Algorithm 2: Efficient Local Greedy Algorithm (ELGA)

Input : (1) k, a positive number (2) w, a target node (3) G, a graph (4) D, the number of simulation times
Output: U, the top-k influential nodes for w

$U \longleftarrow \phi$
while $|U| < k$ **do**
  for *each node* $v \in V \setminus U$ **do**
    $R_w(U \cup \{v\}) = 0$
  end
  for $i = 1$ *to* D **do**
    Create $G_i^*(G, w, p)$
    for *each node* $v \in V \setminus U$ **do**
      $R_w(U \cup \{v\}) += R_w(U \cup \{v\}, G_i^*)$
    end
  end
  for *each node* $v \in V \setminus U$ **do**
    $R_w(U \cup \{v\}) = R_w(U \cup \{v\})/D$
  end
  $U = U \bigcup \arg \max_{v \in V \setminus U} R_w(U \cup \{v\})$
end
Output U

We summarize the improvements in Algorithm 2. The complexity of ELGA can be calculated in detail as follows. In each random process of influence cascade, ELGA first takes $O(m)$ time to induce the graph by simulating the influence cascade on all edges in graph $G_{\overline{w}}$, then takes $O(m_i^*)$ time to compute $R_w(v, G_i^*)$ and $R_w(U, G_i^*)$ by a linear scan of the new induced graph G_i^*, where m_i^* denotes the number of edges in each G_i^* from the original graph $G_{\overline{w}}$. For D repeated simulations with Monte-Carlo, the time complexity of finding k seed nodes is $O(kD \cdot (m + m^*))$, where m^* denotes the average number of edges of all the induced smaller graphs G_i^*, which shows that ELGA achieves better

Algorithm 3: Local Cascade Algorithm (LCA)

Input : (1) k, a positive number (2) w, a target node (3) G, a graph
Output: U, the top-k influential nodes for w

$U \longleftarrow \phi$

for *each node* $v \in V \setminus U$ **do**
  create $L(v, w)$
  $R_w(v) = 1 - \prod_{w_i \in C_w} (1 - R_{w_i}(v) \cdot p_{w_i,w})$
end
while $|U| < k$ **do**
  $U = U \bigcup \arg \max_{v \in V \setminus U} R_w(\{v\})$
end
Output U

efficiency than LGA.

3.3 On-line algorithm

The increasing popularity of many on-line social network sites motivates our online algorithm in this section. We present an efficient and effective heuristic algorithm, referred to as *local cascade algorithm* (LCA for short). Instead of utilizing Monte-Carlo simulation, LCA constructs a local cascade community consists of only the shortest paths between each node in the graph and the target node, then restricts computations [13, 29] within the shortest path community.

In doing so, LCA nearly matches the accuracy of the greedy algorithms, meanwhile significantly reduces the time cost, as we will see later in the experiments.

As shown in Algorithm 3, LCA mainly consists of three steps. First, we construct a local cascade community consisting of only the shortest paths between each node and the target node. Longer paths are viewed as slight influence propagation and omitted in the algorithm. For example, for the node u in Fig. 1, we count only the shortest paths $(u \rightarrow w_2 \rightarrow w)$, $(u \rightarrow w_4 \rightarrow w)$, $(u \rightarrow w_5 \rightarrow w)$ and $(u \rightarrow w_6 \rightarrow w)$ to construct a local cascade community for the influence spread from u to w. Then, We estimate the local influence degree. Instead of using expensive Monte-Carlo method, LCA efficiently estimates the local influence spread by enumerating the shortest simple paths starting from the seed nodes toward the target node. As a result, for a target node w and an arbitrary node $v \in V \setminus w$, the local influence degree $R_w(v)$ in $L(v, w)$ can be estimated as in Eq. (14), where $L(v, w)$ denotes the corresponding local cascade community from v to w.

$$R_w(v) = 1 - \prod_{w_i \in C_w} (1 - R_{w_i}(v) \bullet p_{w_i w}) \tag{14}$$

At the last step, we mine top-k seed nodes based on Eq. (15) for a given target node w in LCA.

$$U = \bigcup_k \arg \max_{v \in V \setminus U} R_w(\{v\}) \tag{15}$$

The time complexity of LCA is $O(m+n\cdot(\overline{L}+k))$, where $\overline{L}$ denotes the average number of edges in all $L(v, w)$. Hence, LCA is much faster than the greedy algorithms LGA and ELGA.

4. EXPERIMENTS

We implement our algorithms using C++ with 3.3Hz CPU and 4GB memory. We use four data sets, one is crawled from a popular social network site in China, *weibo.sina.com*, and the other three are downloaded from Jure Leskovec's Website [30]. We summarize the statistical information of the four data sets in Table 2.

We use other three algorithms as benchmark methods. Moreover, we use Monte-Carlo to choose seed sets in the greedy algorithms, and evaluate their influences on a given target node, as shown in Eq. (11).

- **LDegree algorithm**. Selecting nodes with the highest degree is a popular method in the influence maximization area. In this algorithm, we instead use the global top-k influential nodes as the baseline solution.
- **LRandom algorithm**. In this algorithm, we randomly pick k nodes (expect the target node) in the networks as the local top-k influential nodes for the proposed problem.
- **LND algorithm**. In this algorithm, we simply pick k nodes from the neighbors with highest degree of the target node as the local top-k influential nodes for the proposed problem.

4.1 Parameter Study

4.1.1 Parameter k Study

We evaluate the performance on Dataset-1, by varying k from 1 to 10 with $p = 0.03$ in Figs. 7(a) and 7(b).From the results, we can come to three conclusions:

(1) The local influence degree of seed sets in LGA and ELGA consistently outperforms the other four algorithms. For example, given the target node ID=132, for k=10, the influence degrees on the target node 132 of LGA and ELGA are both 0.3723, while that of LCA, LND, LDegree, LRandom algorithms are 0.3354, 0.3218, 0.1595, and 0.1632 respectively. In other words, LGA and ELGA improve the degree by 0.0369 (or 11.00%) compared to LCA, 0.0505 (or 15.69%) LND, 0.2128 (or 133.42%) LRandom and 0.2091 (or 128.13%) LDegree. This is because LGA and ELGA have performance guarantees as we analyzed before.

(2) The local influence degree of seed sets in LCA is very close to that in LGA and ELGA *w.r.t.* different values of k, which validates the effectiveness of LCA. Meanwhile, LND outperforms LCA in Fig.7(a) for k=1 to 3. This is because LND chooses initial nodes from the perspective of the target node, taking both the factors of distance and degree into consideration. However, as they depend on neighbors of the target node, their superiorities are random and unstable.

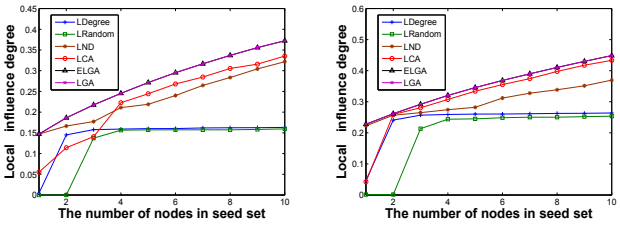
(3) The local influence degrees of seed sets in LDegree and LRandom are always worse than the other four algorithms *w.r.t.* most values of k, which means that LDegree and LRandom only choose initial nodes globally, while losing the local structure *w.r.t.* the target node itself.

Efficiency. The results are shown in Figs. 8(a) and 8(b). We report five algorithms except the LGA greedy algorithm which always consumes the heaviest time. Several conclusions can be made from these results: (1) the running time of ELGA is consistently far worse than the other four algorithms for all values of k. For example, given the target node ID=132, for k=3 and p=0.03, the running time of ELGA is

Table 3: Four data sets for testing

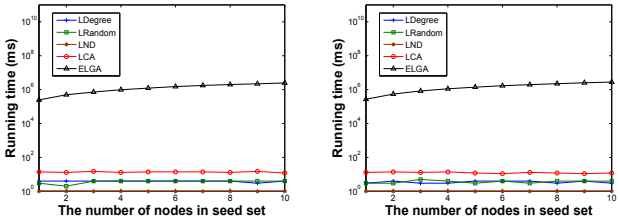
	Dataset-1	Dataset-2 [30]	Dataset-3 [30]	Dataset-4 [30]
Extract from	Weibo.com	Wikipedia.org	Epinions.com	Wikipedia.org
#Description	who-follow-whom network	who-votes-on-whom network	who-trusts-whom network	who-edits-whoes talk page (communication) network
#Nodes	602	7,115	75,879	2,394,385
#Directed Edges	17,595	103,689	508,837	5,021,410
#Average Degree	29.2	14.6	6.7	2.1
#Average Clustering Coefficient	0.3867	0.1409	0.1378	0.0526

720,616ms, while that of LCA, LDegree, LRandom and LND algorithms are 15ms, 4ms, 4ms, and 1ms respectively. Moreover, the efficiency of ELGA is more sensitive *w.r.t.* the increasing number of nodes in the seed set than the other four algorithms. This is because ELGA's time complex is $O(kDm^*)$, which means each round of a new seed node searching process takes $O(Dm^*)$ time. While Monte-Carlo method needs to simulate the random process with sufficient times to obtain very good estimates, D is typically more than 10,000 times, which is very expensive. Therefore, ELGA keeps the computation accuracy at the cost of efficiency. (2) The running time of LCA is very close to the other three benchmark algorithms, for all values of k, which testifies the efficiency of LCA. For example, given a target node ID=132, for k=6 and p=0.03, the running time of LCA is 14ms, while that of LDegree, LRandom and LND are 4ms, 4ms and 1ms respectively. Moreover, the efficiencies of LCA, LND, LDegree and LRandom are nearly the same with the increasing of the number of nodes in the seed set.



(a) (b)

Figure 7: (a)Local influence degree of the target node ID 132 with p=0.03. (b)Local influence degree of the target node ID 125 with p=0.03.

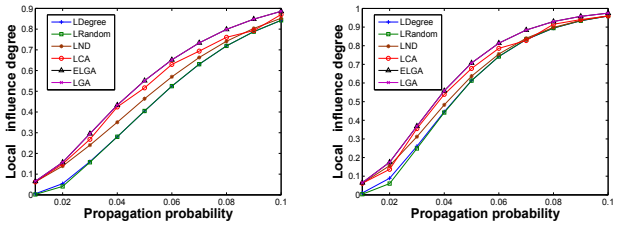


(a) (b)

Figure 8: (a)Running time of target node 132 with p=0.03(b)Running time of target node 125 with p=0.03.

4.1.2 *Parameter p Study*

We evaluate the performance on Dataset-1, by varying p from 0.01 to 0.1 with k=6. The results are shown in Figs. 9(a) and 9(b).



(a) (b)

Figure 9: (a)Local influence degree of the target node 132 with $k = 6$. (b)Local influence degree of the target node 125 with $k = 6$.

(1) The quality of seed sets in LGA and ELGA consistently outperforms the other four algorithms for all values of p. For example, given the node ID=132, for k=6 and p=0.05, the influence degree of LGA and ELGA are both 0.5513, while the evaluation of LCA, LDegree, LRandom and LND are 0.5168, 0.4642, 0.4048 and 0.4046 respectively. In other words, LGA and ELGA improve the influence degree of seed sets by 6.7% (LCA), 18.8% (LDegree), 36.2% (RRandom) and 36.3% (LND).

(2) the quality of seed sets in LCA is very close to that of LGA and ELGA for most values of p, with different target nodes, which testifies the effectiveness of LCA. Meanwhile, the quality performance of LND even performs better than LCA in Fig.9(a) for p=0.09. However, their dominance is randomness and unstable. For example, given the target node ID=132, for k=6 and p=0.06, the influence degree of LND and LCA are 0.5700 and 0.6297 respectively. While given the same node 132 for k=6 and p=0.09, the estimation of LND and LCA are 0.8035 and 0.7931 respectively. In other words, LND drops 9.5% (LCA) for p=0.06, while improves the evaluation of influence degree by 1.3% (LCA) for p=0.09.

(3) The quality of seed sets in LDegree and LRandom are unstable, which always performs worse than the other four algorithms for most values of p. For example, given the node ID=125, for k=6 and p=0.03, the influence degree of LDegree and LRandom are 0.2606 and 0.2488 respectively, while the evaluation of LGA, ELGA, LCA and LND are 0.3686, 0.3686, 0.3554 and 0.3120 respectively. In other words, LDegree and LRandom drop their evaluation influence degrees by 26.7% and 29.3% (LGA/ELGA), 30.0% and 20.3% (L-

CA), and 20.3% and 16.5% (LND).

Efficiency. Figs. 10(a) and 10(b) report the running time performance. Several conclusions can be drawn from these results.

(1) The running time of ELGA is consistently far worse than the other four algorithms for all values of p. For example, given the target node ID=125, for k=6 and p=0.02, the running time of ELGA is 1,326,346ms, while that of LCA, LDegree, LRandom and LND is 13ms, 3ms, 3ms and 1ms respectively. Meanwhile, the efficiency performance of ELGA is more sensitive to the increasing of the propagation probability than the other four algorithms. As the number of activated nodes in the graph is growing with the propagation probability, it further causes the increasing in the scale of the new induced graph G^* in each iteration, and results in the efficiency-reduction of ELGA.

(2) The running time of LCA is very close to that from the other four benchmark algorithms, for all values of p, which testifies the efficiency of LCA. For example, given the target node ID=125, for k=6 and p=0.04, the running time of LCA is 15ms, while that of LDegree, LRandom and LND are 4ms, 4ms and 1ms respectively. Meanwhile, the efficiencies of LCA, LND, LDegree and LRandom are almost unchanged with the increasing of the propagation probability. This is because the propagation probability in the seed set has little connection with efficiency performance.

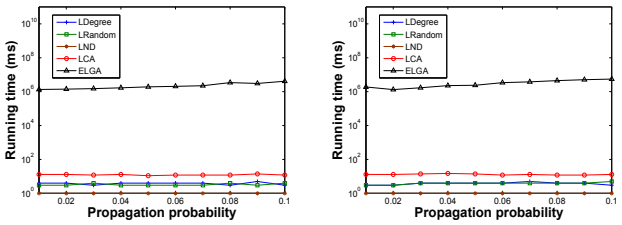


Figure 10: (a)Running time for the target node ID 132 with $k = 6$ (b)Running time for the target node ID 125 with $k = 6$.

4.2 Scalability Study

We evaluate the LCA algorithm under different network scales (from Dataset-1 to Dataset-4). Fig.11(a) shows the efficiency results with k varying from 1 to 10 and p=0.03. Fig.11(b) shows the efficiency results with p varying from 0.01 to 0.1 and k=6. We can observe that LCA has good adaptability under different networks, it can handle large-scale data sets with different values of parameters. For example, in Fig. 11(b), for p varying from 0.05 to 0.07, the running time on Dataset-4 is 93ms (or 0.7%) and 15ms(or 0.1%), which can be viewed as a relatively stable process.

5. RELATED WORK

Given a social network with influence weights, the global influence maximization aims to identify a subset of seed nodes (i.e., leaders), through which the influence can be spread to the largest number of users in the social network.

The pioneer work in global influence maximization, to the best of our knowledge, can be traced back to Domingos and

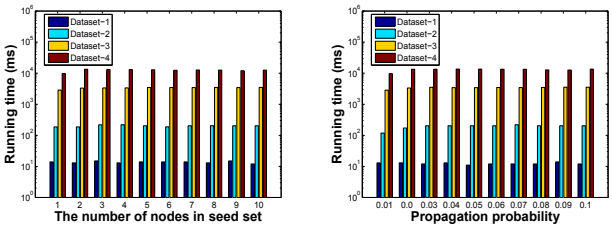


Figure 11: (a)Running time with $p = 0.03$ (b)Running time with $k = 6$

Richardson [7, 8]. Then, Kempe et al. [1] formulated the influence maximization problem as a discrete optimization problem, and provided algorithms with provable approximation guarantees.

For algorithms, most recent work [3, 9–16, 21, 26] focus on the efficiency of the influence maximization problem, due to expensive computation on large-scale social networks. These research can be divided into two complementary directions. One direction is greedy algorithms and their extensions with provable approximation guarantees [3, 10–12, 16, 21, 26]. For example, Leskovec et al. [3, 11] and Goyal et al. [16] exploited the sub-modularity property of problem objective to reduce the number of evaluations on the influence spread of nodes. Kimura et al. [10] estimated all quantities on the basis of bond percolation and graph theory. Chen et al. [11] removed the edges that not contribute to propagation from the original graph, and performed the influence diffusion on a smaller graph. Wang et al. [12] found influential nodes based on communities detecting and selecting. Barbieri et al. [26] studied social influence from the topic modeling perspective.

The other direction is to develop efficient heuristic algorithms on large-scale networks [9, 11, 13–15]. For example, Kimura and Saito [9] proposed two shortest-path based influence cascade models for efficient estimation. Chen et al. [11] found influential nodes under the assumption that the influence spread increases with the degree of nodes, and gained efficiency by restricting computations on the local local influence communities of nodes [13, 14]. Goyal et al. [15] computed the spread by exploring simple paths in the neighborhood. Different from greedy algorithms, heuristics has high efficiency, but without provable performance guarantee.

However, existing work on global influence maximization cannot be directly transplanted to our local optimization problem. Therefore, we present several local near-optimal algorithms with approximation guarantee.

6. CONCLUSION AND FUTURE WORK

We have studied a new problem of personalized (local) influence maximization on social networks. We have provided a random function, with lower variance guarantee, to randomly simulate our objective function. Then, we have presented efficient algorithms with approximation guarantees. Experimental results have validated the performance of the proposed algorithms.

There are some interesting problems in the future. First, our local influence maximization model is based on the IC

model. How to model the problem under other cascade models (e.g., the linear threshold model) needs further study. Second, with the availability of big social network data generated by online shopping, advertising and instant messaging, how to identify the most influential users for a given target user across multiple sources is also a challenge.

7. ACKNOWLEDGMENTS

This work is supported by the Research Project of Institute of Information Engineering of Chinese Academy of Sciences (No. Y3Z0062101), the National Science Foundation of China (No. 61003167), the National High Technology Research and Development Program of China (No. 2011AA010703 and 2011AA01A103), the Strategic Leading Science and Technology Projects of Chinese Academy of Sciences (No. XDA06030200).

8. REFERENCES

[1] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *In Proc. of KDD*, pages 137–146, 2003.

[2] Xiaodan Song, Belle L.Tseng, Ching-Yung Lin, and Ming-Ting Sun. Personalized recommendation driven by information flow. *In Proc. of SIGIR*, pages 509–516, 2006.

[3] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne Vanbriesen, and Natalie Glance. Cost-effective outbreak detection in networks. *In Proc. of KDD*, pages 420–429, 2007.

[4] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: Finding topic-sensitive influential twitterers. *In Proc. of WSDM*, pages 261–270, 2010.

[5] Eytan Bakshy, Winter A. Mason, Jake M. Hofman, and Duncan J. Watts. Everyone's an influencer: Quantifying influence on twitter. *In Proc. of WSDM*, pages 65–74, 2011.

[6] Elchanan Mossel and Sebastien Roch. On the submodularity of influence in social networks. *In Proc. of STOC*, pages 128–134, 2007.

[7] Domingos Pedro and Richardson Matt. Mining the network value of customers. *In Proc. of KDD*, pages 57–66, 2001.

[8] Richardson Matthew and Domingos Pedro. Mining knowledge-sharing sites for viral marketing. *In Proc. of KDD*, pages 61–70, 2002.

[9] Masahiro Kimura and Kazumi Saito. Tractable models for information diffusion in social networks. *In Proc. of PKDD*, pages 259–271, 2006.

[10] Masahiro Kimura, Kazumi Saito, and Ryohei Nakano. Extracting influential nodes for information diffusion on a social network. *In Proc. of AAAI*, pages 1371–1376, 2007.

[11] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. *In Proc. of KDD*, pages 199–208, 2009.

[12] Yu Wang, Gao Cong, Guojie Song, and Kunqing Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. *In Proc. of KDD*, pages 1039–1048, 2010.

[13] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. *In Proc. of KDD*, pages 1029–1038, 2010.

[14] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. *In Proc. of ICDM*, pages 88–97, 2010.

[15] Amit Goyal, Wei Lu, and Laks V.S Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. *In Proc. of ICDM*, pages 211–220, 2011.

[16] Amit Goyal, Wei Lu, and Laks V.S Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. *In Proc. of WWW*, pages 47–48, 2011.

[17] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the spread of misinformation in social networks. *In Proc. of WWW*, pages 665–674, 2011.

[18] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. *In Proc. of WINE*, pages 306–311, 2007.

[19] Smriti Bhagat, Amit Goyal, and Laks V.S Lakshmanan. Maximizing product adoption in social networks. *In Proc. of WSDM*, pages 603–612, 2012.

[20] Tim Carnes, Chandrashekhar Nagarajan, Stefan M. Wild, and Van Zuylen Ankee. Maximizing influence in a competitive social network: A follower's perspective. *In Proc. of ICEC*, pages 351–360, 2007.

[21] Jure Leskovec. *Dynamics of Large Networks.* Addison-Wesley Publishing Company, 2008.

[22] Seth A. Myers and Jure Leskovec. Clash of the contagions: Cooperation and competition in information diffusion. *In Proc. of ICDM.*

[23] Julian McAuley and Jure Leskovec. Learning to discover social circles in ego networks. *In Proc.c of NIPS.*

[24] Seth A. Myers, Chenguang Zhu, and Jure Leskovec. Information diffusion and external influence in networks. *In Proc.c of KDD.*

[25] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *TKDD.*

[26] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. Topic-aware social influence propagation models. *In Proc. of ICDM*, pages 81–90, 2012.

[27] David Williams. Probability with martingales. *Cambridge University Press.*

[28] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, pages 265–294, 1978.

[29] Jure Leskovec, Ajit Singh, and Jon Kleinberg. Patterns of influence in a recommendation network. *In Proc. of PAKDD*, pages 380–389, 2006.

[30] http://snap.stanford.edu/data/index.html.