2013 IEEE 13th International Conference on Data Mining

UBLF: An Upper Bound Based Approach to Discover Influential Nodes in Social Networks

Chuan Zhou*, Peng Zhang*, Jing Guo†*, Xingquan Zhu‡ and Li Guo*
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
‡Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA
Email:{zhouchuan,zhangpeng}@iie.ac.cn, guojing@nelmail.iie.ac.cn, xzhu3@fau.edu, guoli@iie.ac.cn

***Abstract*—Influence maximization, defined as finding a small subset of nodes that maximizes spread of influence in social networks, is NP-hard under both Linear Threshold (LT) and Independent Cascade (IC) models, where a line of greedy/heuristic algorithms have been proposed. The simple greedy algorithm [14] achieves an approximation ratio of $1-1/e$. The advanced CELF algorithm [16], by exploiting the submodular property of the spread function, runs 700 times faster than the simple greedy algorithm on average. However, CELF is still inefficient [4], as the first iteration calls for N times of spread estimations (N is the number of nodes in networks), which is computationally expensive especially for large networks. To this end, in this paper we derive an upper bound function for the spread function. The bound can be used to reduce the number of Monte-Carlo simulation calls in greedy algorithms, especially in the first iteration of initialization. Based on the upper bound, we propose an efficient *Upper Bound based Lazy Forward* algorithm (UBLF in short), by incorporating the bound into the CELF algorithm. We test and compare our algorithm with prior algorithms on real-world data sets. Experimental results demonstrate that UBLF, compared with CELF, reduces more than 95% Monte-Carlo simulations and achieves at least $2-5$ times speed-raising when the seed set is small.**

***Keywords*-Influence Maximization, Social Networks, Independent Cascade Model, Greedy Algorithms.**

I. Introduction

Influence maximization is one of the fundamental problems in social networks, which has received significant attention in recent years [1], [2]. The motivation of influence maximization comes from viral marketing, where the target is to give free or price-discounted samples of a product to fine-selected individuals, such that through the word of mouth effect, the propagation can result in maximum number of adoptions of the product.

The seminal work, by Kempe, Kleinberg and Tardos [14], first formulates influence maximization as a discrete optimization problem: Given a directed social graph with users as nodes, edge weights reflecting influence between users and a budget/threshold number k, finding k nodes in the graph, such that by activating these nodes, the expected spread of the influence can be maximized, based on a given stochastic influence propagation model.

Two popularly used stochastic influence propagation models are the *Independent Cascade* (IC) and *Linear Threshold* (LT) models [14]. In both models, at any time step, a user is represented as a binary variable with either active (an adopter of the product) or inactive status, and influence propagates until no more users can become active. The major difference between the two models is the way of an active user propagating its influence to the neighbors: For the IC model when an inactive user becomes active at a time step t, it has exactly one chance to independently activate its currently inactive neighbors at the next time step $t+1$; while in the LT model, the sum of incoming edge weights on any node is assumed to be at most 1, every user chooses an activation threshold uniformly at random from $[0,1]$, and at any time step, a node becomes activated if the sum of incoming edge weights from the active neighbors exceeds the threshold.

Influence maximization under both IC and LT models is NP-hard, and the spread function is monotone and submodular [14]. A set function $f: 2^V \to \mathbb{R}^+$ is monotone, if $f(S) \leq f(T)$ whenever $S \subseteq T \subseteq V$. The set function is submodular, if $f(S\cup\{w\}) - f(S) \geq f(T\cup\{w\}) - f(T)$ for all $S \subseteq T$ and $w \in V\backslash T$. Intuitively, submodularity indicates that f has diminishing margin returns when adding more nodes into the set.

Exploiting these two properties, Kempe et al. [14] presented a simple greedy algorithm which repeatedly chooses the node with the maximum marginal gain and adds it to the seed set, until the budget k is reached. However, computing exact marginal gain (or exact expected spread) under both the IC and LT models is $\#P$-hard [5], [6]. In practice, it is usually estimated by running Monte Carlo (MC) simulations. The simple greedy algorithm can approximate the solution within a factor of $(1-1/e-\epsilon)$ for any $\epsilon > 0$.

Unfortunately, the simple greedy algorithm suffers from two major sources of inefficiency. (I) The MC simulations that run sufficiently many times (typically 10,000) to obtain an accurate estimate of spread, has been proved computationally expensive especially for large networks. (II) The greedy algorithm calls for $O(kN)$ iterations at the spread estimation step, where k is the size of initially picked seed

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.55

set, and N is the number of nodes. When N is large, the efficiency of the algorithm is unsatisfactory.

Considerable work has been conducted to tackle the above two limitations. To address the first limitation, many heuristic solutions have been proposed to improve the efficiency of seed selection, *e.g.*, DegreeDiscount [4], MIA [5], DAG [6], SIMPATH [11], ShortestPath [15] and SPIN [18]. The heuristic algorithms proposed in these works can reduce computational cost in orders of magnitude, with competitive results of the influence spread level. However, none of them has a theoretical guarantee on the reliability of the results. In other words, it is unknown how far these heuristic solutions approximate the optimal solution. One can only borrow the simple greedy algorithm as the benchmark for performance testing.

To tackle the second limitation, a representative work, by Leskovec et al. [16], exploited the submodular property of the objective function, and proposed a Cost-Effective Lazy Forward selection (CELF) algorithm. The algorithm can significantly reduce the number of MC simulation calls in the simple greedy algorithm. The principle behind is that the marginal gain of a node in the current iteration cannot be more than that in previous iterations, and thus the number of spread estimation calls can be greatly pruned. Leskovec et al. [16] reported that CELF improves the running time of the simple greedy algorithm by up to 700 times. Following the same logic, Goyal et al. proposed CELF++ [10], an extension of CELF, that further reduces the number of spread estimation calls, leading to $35\% - 55\%$ faster than CELF.

Although CELF significantly improves the the simple greedy algorithm, its running time is still quite slow on large networks [4]. In particular, in the initialization step, CELF needs to estimate the spread using Monte-Carlo for each node in a network, resulting in N times of Monte-Carlo calls (N is the total number of nodes in the network), which is time-consuming, especially when the network is very large. The limitation leads to a rather fundamental question that, *can we derive an upper bound of spreads which can be used to prune unnecessary spread estimations (Monte-Carlo calls) in the CELF algorithm*? To the best of our knowledge, there is no work in the literature that mathematically discuss the upper bound properties of the spread function.

Motivated by the above question, in this paper we derive an upper bound for greedy algorithms in influence maximization problem. Based on the bound, we propose a new greedy algorithm *Upper Bound based Lazy Forward* (UBLF for short), which outperforms the original CELF algorithm. The main contribution of the paper is threefold:

1) We derive an upper bound for spread $\sigma_I(S)$ whose exact expected estimation under the IC model is $\#P$-hard.
2) We propose, based on the upper bound, an efficient UBLF algorithm to discover the influential nodes in social networks.
3) We conduct extensive experiments on real-world data sets to demonstrate the performance of the proposed UBLF algorithm.

The rest of the paper is organized as follows. In Section II, we briefly review IC model and the simple greedy algorithm. In Section III, we derive the upper bound for the spread function $\sigma_I(S)$ and the UBLF algorithm. In Section IV, we verify the performance of our algorithm by experiments. We survey major related work in Section V and conclude the paper in Section VI. Table 1 outlines the key variables used in the paper.

II. IC Model and Greedy Algorithm

Consider a directed graph $G = (V, E)$ with N nodes in V and edge labels $pp : E \to [0, 1]$. For each edge $(u, v) \in E$, $pp(u, v)$ denotes the propagation probability that v is activated by u through the edge. If $(u, v) \notin E, pp(u, v) = 0$. Let $Par(v)$ be the set of parent nodes of v, *i.e.*,

$$Par(v) := \big\{u \in V,\ (u, v) \in E\big\}.$$

Given an initially activated set $S \subseteq V$, the independent cascade (IC) model works as follows. Let $S_t \subseteq V$ be the set of nodes that are activated at step $t \geq 0$, with $S_0 = S$. Then, at step $t + 1$, each node $u \in S_t$ may activate its out-neighbors $v \in V \backslash \cup_{0 \leq i \leq t} S_i$ with an independent probability of $pp(u, v)$, where $\cup_{0 \leq i \leq t} S_i := S_0 \cup S_1 \cup \cdots \cup S_t$. Thus, a node $v \in V \backslash \cup_{0 \leq i \leq t} S_i$ is activated at step $t + 1$ with the probability

$$1 - \prod_{u \in S_t \cap Par(v)} \big(1 - pp(u, v)\big) \tag{1}$$

where the subscript $u \in S_t \cap Par(v)$ means that node u, a parent node of v, is activated at step t. If node v is successfully activated, it is added into the set S_{t+1}. The process ends at a step τ with $S_\tau = \varnothing$. Obviously, the propagation process has $N - |S|$ steps at most, as there are at most $N - |S|$ nodes outside the seed set S. Let $S_{\tau+1} = \varnothing, \cdots, S_{N-|S|} = \varnothing$, if $\tau < N - |S|$. Note that each activated node only has one chance to activate its out-neighbors at the step right after itself is activated, and each node stays activated once it is activated by others.

In the IC model, the influence spread of a seed set S, which is the expected number of activated nodes by S, is denoted by $\sigma_I(S)$ as follow,

$$\sigma_I(S) := \mathbb{E}^S\Big[\big|\bigcup_{t=0}^{N-|S|} S_t\big|\Big] \tag{2}$$

where $\mathbb{E}^S$ is the expectation operator with set S, the subscript 'I' denotes the IC model, $\bigcup_{t=0}^{N-|S|} S_t := S_0 \cup \cdots \cup S_{N-|S|}$ is the sets of nodes activated in all $N - |S| + 1$ steps.

Table I
MAJOR VARIABLES IN THE PAPER

Variables	Descriptions
$G = (V, E)$	social network G with node set V edge set E
N	number of nodes in the network G
S	initial seed set
S_t	set of activated nodes at step t
$\lvert S \rvert$	number of nodes in S
k	number of seeds to be selected
$Par(v)$	set of parents of node v
$\mathbb{P}^S$	probability measure with the seed set S
$\mathbb{E}^S$	expectation operator with the seed set S
Π_t^S	row vector with probabilities as in Eq. (9)
PP	N by N propagation probability matrix
$\mathbf{1}$	column vector with all elements being 1

The above equation provides us convenience to treat the global influence function, $\sigma_I(S)$, as a summation of locally activated node sets S_t ($1 \leq t \leq N - |S|$), as we will see in *Proposition 1* in the next section.

The influence maximization problem, under the IC model, is to find a subset $S^* \subseteq V$ such that $|S^*| = k$ and $\sigma_I(S^*) = \max\big\{\sigma_I(S) \bigm| |S| = k, S \subseteq V\big\}$, *i.e.*,

$$S^* = \arg \max_{|S|=k, S \subseteq V} \sigma_I(S) \tag{3}$$

where k is a given parameter. The problem, as proved in a previous work [14], is NP-hard, and a constant-ratio approximation algorithm is feasible.

In the work [14], [17], it is shown that the objective function $\sigma_I(S)$ in Eq.(3) has the submodular and monotone properties [14] with $\sigma_I(\varnothing) = 0$. Thus, the problem in Eq.(3) can be approximated by the greedy algorithm as shown in Algorithm 1. Theoretically, a non-negative real valued function f on subsets of V is submodular, if $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ for all $v \in V$ and $S \subseteq T \subseteq V$. Thus, f has diminishing marginal return. Moreover, f is monotone, if $f(S) \leq f(T)$ for all $S \subseteq T$. For any submodular and monotone function f with $f(\varnothing) = 0$, the problem of finding a set S of size k that maximizes $f(S)$ can be approximated by the greedy algorithm in Algorithm 1. The algorithm iteratively selects a new seed u that maximizes the incremental change of f, to be included into the seed set S, until k seeds are selected. It is shown in [19] that the algorithm guarantees the approximation ratio $f(S)/f(S^*) \geq 1 - 1/e$, where S is the output of the greedy algorithm and S^* is the optimal solution.

In Algorithm 1, an important issue is that there is no efficient way to compute $\sigma_I(S)$ given a set S. Kempe et al. [14] run Monte-Carlo simulations of the propagation model for $10,000$ trials to obtain an accurate estimate of the expected spread, leading to very expensive computation cost. Chen et al. [5] pointed out that computing $\sigma_I(S)$ is actually #P-hard, by showing a reduction from the counting problem of s-t connectness in a graph.

Algorithm 1: Greedy(k,f)

```
initial $S = \varnothing$
**for** $i = 1$ to $k$ **do**
    select $u = \arg\max_{w \in V \setminus S}(\sigma_I(S \cup \{w\}) - \sigma_I(S))$
    $S = S \cup \{u\}$
**end for**
output $S$
```

Based on the above observations, in order to improve the efficiency of Algorithm 1, one can either reduce the number of times calling Monte-Carlo simulations in computing $\sigma_I(S)$, or develop advanced heuristic algorithms which reduce the number of iterations without accuracy guarantees.

III. ANALYSIS AND APPROACHES

In this part, we aim to derive an upper bound of $\sigma_I(S)$, as the exact computation of $\sigma_I(S)$ is #P-hard [5]. To the best of our knowledge, we are the first to discuss the upper bound of the influence spread in the literature. The upper bound provides us a new view to design efficient algorithms in the field of influence maximization.

A. Upper bound of $\sigma_I(S)$

Before introducing the bound in Theorem 1 and Corollary 1, we introduce two preparations first. Let $\mathbb{P}^S(v \in S_t)$ denote the probability that node v becomes activated at step t under the seed S, we have the first preparation as follow,

Proposition 1: For $S \subseteq V$, the spread $\sigma_I(S)$ under IC model can be calculated as

$$\sigma_I(S) = \sum_{t=0}^{N-|S|} \sum_{v \in V} \mathbb{P}^S(v \in S_t). \tag{4}$$

Proof: Note that the random sets $S_0, S_1, \ldots, S_{N-|S|}$ are pairwise disjoint, we have

$$\begin{aligned}
\sigma_I(S) &= \mathbb{E}^S\Big[\Big|\bigcup_{t=0}^{N-|S|} S_t\Big|\Big] = \mathbb{E}^S\Big[\sum_{t=0}^{N-|S|} |S_t|\Big] \\
&= \sum_{t=0}^{N-|S|} \mathbb{E}^S\Big[|S_t|\Big] = \sum_{t=0}^{N-|S|} \mathbb{E}^S\Big[\sum_{v \in V} I_{S_t}(v)\Big] \\
&= \sum_{t=0}^{N-|S|} \sum_{v \in V} \mathbb{E}^S\big[I_{S_t}(v)\big] \\
&= \sum_{t=0}^{N-|S|} \sum_{v \in V} \mathbb{P}^S(v \in S_t)
\end{aligned}$$

where $I_{S_t}(v)$ is an binary indicative function, if $v \in S_t$, $I_{S_t}(v) = 1$; otherwise, $I_{S_t}(v) = 0$. □

Proposition 1 reveals that we can treat the global influence measure $\sigma_I(S)$ as a summation of all $N-|S|$ propagation steps of local probabilities $\big\{\mathbb{P}^S(v \in S_t) : t \geq 0, v \in V\big\}$.

Based on Proposition 1, a following question is, *what is the relationship between two sets,*

$$\big\{\mathbb{P}^S(v \in S_t) : v \in V\big\}$$

and

$$\big\{\mathbb{P}^S(v \in S_{t-1}) : v \in V\big\}.$$

Proposition 2: For $t = 1, 2, \ldots, N-|S|$ and $v \in V$, we have the following inequation

$$\mathbb{P}^S(v \in S_t) \leq \sum_{u \in V} \mathbb{P}^S\big(u \in S_{t-1}\big) pp(u,v). \tag{5}$$

Proof: For $t = 1, 2, \ldots, N-|S|$, by the definition of conditional expectation and IC model, it follows that

$$\begin{aligned}
& \mathbb{P}^S(v \in S_t) \\
= \ & \mathbb{E}^S\big[\mathbb{P}^S(v \in S_t | S_0, \cdots, S_{t-1})\big] \\
= \ & \mathbb{E}^S\Big[I_{\{v \notin \cup_{i=0}^{t-1} S_i\}} \cdot \big(1 - \prod_{u \in S_{t-1}} (1 - pp(u,v))\big)\Big] \\
\leq \ & \mathbb{E}^S\Big[I_{\{v \notin \cup_{r=0}^{t-1} S_r\}} \cdot \big(\sum_{u \in S_{t-1}} pp(u,v)\big)\Big] \\
= \ & \mathbb{E}^S\Big[I_{\{v \notin \cup_{r=0}^{t-1} S_r\}} \cdot \big(\sum_{u \in V} I_{\{u \in S_{t-1}\}} pp(u,v)\big)\Big] \\
= \ & \mathbb{E}^S\Big[\sum_{u \in V} I_{\{v \notin \cup_{r=0}^{t-1} S_r, u \in S_{t-1}\}} \cdot pp(u,v)\Big] \\
= \ & \sum_{u \in V} \mathbb{P}^S\big(v \notin \cup_{r=0}^{t-1} S_r, u \in S_{t-1}\big) pp(u,v) \\
= \ & \sum_{u \in V} \mathbb{P}^S\big(v \notin \cup_{r=0}^{t-1} S_r \big| u \in S_{t-1}\big) \mathbb{P}\big(u \in S_{t-1}\big) pp(u,v) \\
\leq \ & \sum_{u \in V} \mathbb{P}^S\big(u \in S_{t-1}\big) pp(u,v).
\end{aligned}$$

In the above derivation, $\{v \notin \cup_{i=0}^{t-1} S_i\}$ means that node v does not belong to any set of $S_0, \cdots, S_{t-1}$. The first '=' stems from the conditional expectation, the second '=' is from the assumption of IC model, the first '$\leq$' comes from the fact that

$$1 - \prod_{i=1}^{n} (1 - x_i) \leq \sum_{i=1}^{n} x_i \tag{6}$$

and the second '$\leq$' is due to

$$\mathbb{P}^S\big(v \notin \cup_{r=0}^{t-1} S_r \big| u \in S_{t-1}\big) \leq 1. \tag{7}$$

□

Proposition 2 clearly identifies the ordering relationship between two adjacent elements in the series $\mathbb{P}^S(v \in S_0), \cdots, \mathbb{P}^S(v \in S_t), \cdots, \mathbb{P}^S(v \in S_{N-|S|})$.

Now we simplify the results in Propositions 1 and 2 into the form of matrix. Let PP be the propagation probabilities matrix with the (u,v) position's element being $pp(u,v)$. For $t = 0, 1, 2, \ldots, N-|S|$, denote the row vector

$$\Pi_t^S = (\pi_t^S(v)) \tag{8}$$

as the probabilities of nodes being activated at step t, *i.e.*,

$$\pi_t^S(v) := \mathbb{P}^S(v \in S_t). \tag{9}$$

Then, Proposition 1 can be rewritten as

$$\sigma_I(S) = \sum_{t=0}^{N-|S|} \Pi_t^S \cdot \mathbf{1} \tag{10}$$

where $\mathbf{1}$ is a column vector with all elements being 1 and the notation '$\cdot$' is matrix multiplication. Likewise, Proposition 2 can be rewritten as

$$\Pi_t^S \leq \Pi_{t-1}^S \cdot PP \tag{11}$$

where PP denotes the propagation probability matrix. Then we have Theorem 1 as follow,

Theorem 1: The upper bound of $\sigma_I(S)$ is

$$\sigma_I(S) \leq \sum_{t=0}^{N-|S|} \Pi_0^S \cdot PP^t \cdot \mathbf{1}. \tag{12}$$

Proof: By the iteration in Eq.(11), we have

$$\Pi_t^S \leq \Pi_0^S \cdot PP^t \tag{13}$$

where

$$\Pi_0^S = (\pi_0^S(v)) = \begin{cases} 1, & \text{if } v \in S \\ 0, & \text{if } v \notin S \end{cases}$$

By incorporating Eq.(13) into Eq. (10), we have the following inequation,

$$\sigma_I(S) = \sum_{t=0}^{N-|S|} \Pi_t^S \cdot \mathbf{1} \leq \sum_{t=0}^{N-|S|} \Pi_0^S \cdot PP^t \cdot \mathbf{1}.$$

Note that here $\Pi_0^S \cdot PP^t \cdot \mathbf{1}$ is a real number after matrix calculation. □

Based on Eq.(12) in Theorem 1, one may easily raise the following two questions,

- The function $\sigma_I(S)$ is bounded by a summation of series $\sum_{t=0}^{N-|S|} \Pi_0^S \cdot PP^t \cdot \mathbf{1}$, if we relax the series to $\sum_{t=0}^{\infty} \Pi_0^S \cdot PP^t \cdot \mathbf{1}$, then in what condition the series will be convergent?
- If the relaxed series is convergent, what's the limit of convergence, *i.e.*, $\sum_{t=0}^{\infty} \Pi_0^S \cdot PP^t \cdot \mathbf{1} = ?$.

In the sequel, we derive Corollary 1 to answer the two questions.

Corollary 1: If the propagation probability satisfies the conditions

$$\max_{v}\sum_{u} pp(u,v) < 1 \quad \text{or} \quad \max_{u}\sum_{v} pp(u,v) < 1, \tag{14}$$

then the series in Eq.(12) is convergent, and the limit of convergence exists,

$$\sigma_I(S) \leq \Pi_0^S \cdot (E - PP)^{-1} \cdot \mathbf{1} \tag{15}$$

where E is an unit matrix and $(E-PP)^{-1}$ is the inverse of $(E-PP)$.

Proof: By matrix analysis [13], condition (14) implies

$$\| PP \|_1 < 1 \quad \text{or} \quad \| PP \|_\infty < 1,$$

which ensures the convergence of matrix series $\sum_{t=0}^{\infty} PP^t$ with the limit $(E-PP)^{-1}$. Hence we have

$$\begin{aligned} \sigma_I(S) &\leq \sum_{t=0}^{N-|S|} \Pi_0^S \cdot PP^t \cdot \mathbf{1} \\ &\leq \sum_{t=0}^{\infty} \Pi_0^S \cdot PP^t \cdot \mathbf{1} \\ &= \Pi_0^S \cdot \left(\sum_{t=0}^{\infty} PP^t \right) \cdot \mathbf{1} \\ &= \Pi_0^S \cdot (E-PP)^{-1} \cdot \mathbf{1} \end{aligned}$$

where the first '$\leq$' is from Theorem 1. □

Because the inverse $(E-PP)^{-1}$ may be intractable when the size of network is very large, we adopt the following method to calculate $(E-PP)^{-1} \cdot \mathbf{1}$. For $t \geq 0$, we denote the column vector $\mathbf{a}_t := PP^t \cdot \mathbf{1}$, so we have

$$(E-PP)^{-1} \cdot \mathbf{1} = \sum_{t=0}^{\infty} PP^t \cdot \mathbf{1} = \sum_{t=0}^{\infty} \mathbf{a}_t$$

where $\mathbf{a}_{t+1} = PP \cdot \mathbf{a}_t$. With this iteration, we sum up $\mathbf{a}_0, \mathbf{a}_1, \cdots$ until some $\mathbf{a}_n$ with L_1-norm less than 10^{-6}. This transformation saves memory space during calculation, as it only stores vectors instead of matrixes in the memory.

Based on Eq.(14), we can observe that the matrix series converges on condition that either the total influence to any node is less than 1, or the total influence diffused by any node is less than 1. In real-world social networks, the propagation probability is often very small. Thus, Condition (14) usually stands.

Now we use an example to explain the bound calculation.

Example 1: Given a graph G, as shown in Fig. 1, with propagation probability matrix in Eq. (16),

$$PP = \begin{pmatrix} 0 & 0.2 & 0.1 & 0 \\ 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0.2 \\ 0.1 & 0 & 0 & 0 \end{pmatrix} \tag{16}$$

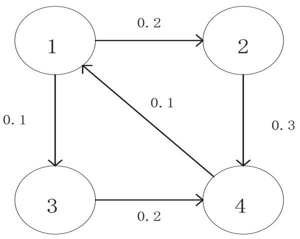


Figure 1. An illustration of the upper bound calculation.

we have

$$\begin{aligned} & (E-PP)^{-1} \cdot \mathbf{1} \\ = & \begin{pmatrix} 1 & -0.2 & -0.1 & 0 \\ 0 & 1 & 0 & -0.3 \\ 0 & 0 & 1 & -0.2 \\ -0.1 & 0 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\ = & \begin{pmatrix} 1.3911 \\ 1.3417 \\ 1.2278 \\ 1.1391 \end{pmatrix} \end{aligned}$$

Based on Corollary 1, the upper bound of spread $\sigma_I(S)$ with the seed set $S = \{\textcircled{2}, \textcircled{4}\}$ can be calculated as follow,

$$\begin{aligned} \sigma_I\left(\textcircled{2}, \textcircled{4}\right) &\leq \Pi_0^{(\textcircled{2},\textcircled{4})} \cdot (E-PP)^{-1} \cdot \mathbf{1} \\ &= (0\ 1\ 0\ 1) \cdot \begin{pmatrix} 1.3911 \\ 1.3417 \\ 1.2278 \\ 1.1391 \end{pmatrix} = 2.4808 \end{aligned}$$

□

B. UBLF algorithm

The Cost-Effective Lazy Forward (CELF) algorithm, proposed by Leskovec *et al.* [16], exploited the submodular property to improve the simple greedy algorithm. The idea is that the marginal gain of a node in the current iteration cannot be more than that in previous iterations, and thus the number of spread estimations can be significantly reduced.

However, CELF demands N spread estimations to establish the initial bounds of marginal increments, which is time expensive on large graphs. In our Upper Bound based Lazy Forward (UBLF) algorithm, we use the derived upper bound to further reduce the number of spread estimations in the initialization step. By doing so, the nodes will be all ranked by their upper bound scores, which can potentially reduce the computational cost of the original CELF algorithm. We use Example 2 for illustration.

Example 2: We still use the network in Fig. 1 for explanation. **The goal here is to find the top-1 node with maximal influence.** For a specific node ①, according to

Corollary 1, we have its upper bound as follow,

$$\begin{aligned}\sigma_I\big(\textcircled{1}\big) &\leq \Pi_0^{\textcircled{1}} \cdot (E-PP)^{-1} \cdot \mathbf{1} \\ &= (1\ 0\ 0\ 0) \cdot \begin{pmatrix} 1.3911 \\ 1.3417 \\ 1.2278 \\ 1.1391 \end{pmatrix} = 1.3911\end{aligned}$$

By the same logic, we can obtain

$$\sigma_I\big(\textcircled{2}\big) \leq 1.3417,\ \sigma_I\big(\textcircled{3}\big) \leq 1.2278,\ \sigma_I\big(\textcircled{4}\big) \leq 1.1391$$

Obviously, the upper bound of $\sigma_I\big(\textcircled{1}\big)$, 1.3911 , is the largest in the graph. Thus, we use Monte-Carlo simulation to estimate $\sigma_I\big(\textcircled{1}\big)$ (or explicitly calculate it due to the simple structure), and get

$$\sigma_I\big(\textcircled{1}\big) = 1.3788$$

Now, we can observe that 1.3788 is already larger than the upper bounds of $\sigma_I\big(\textcircled{2}\big)$, $\sigma_I\big(\textcircled{3}\big)$ and $\sigma_I\big(\textcircled{4}\big)$. Thus, we do not need extra Monte-Carlo simulations to estimate the other three nodes, and node $\textcircled{1}$ is the node with the maximal influence in the graph.

By introducing the upper bounds, we can greatly reduce the number of Monte-Carlo simulation calls. In Example 2, we use only one Monte-Carlo simulation call, while in the CELF [16] algorithm, we need four Monte-Carlo simulation calls. □

We summarize the UBLF algorithm in Algorithm 2.

Algorithm 2: UBLF

```
Input: the propagation probability matrix $PP$ of a
        graph $G = (V, E)$, a budget $k$
Output: The most influential set $S$ with $k$ nodes
initial $S \leftarrow \varnothing$ and $\delta \leftarrow (E - PP)^{-1} \cdot \mathbf{1}$
for $i = 1$ to $k$ do
    set $I(v) \leftarrow 0$ for $v \in V \backslash S$
    while TRUE do
    {
      $u \leftarrow \arg\max_{v \in V \backslash S} \delta_v$
      if $I(u) = 0$
         $\delta_u \leftarrow MC(S \cup \{u\}) - MC(S)$
         $I(u) \leftarrow 1$
      end if
      if $\delta_u \geq \max_{v \in V \backslash (S \cup \{u\})} \delta_v$
         $S \leftarrow S \cup \{u\}$
         break
      end if
    }
end for
output $S$
```

In Algorithm 2, the column vector, $\delta = \big\{\delta_u\big\}$, denotes upper bounds of marginal increments under the current seed set S, *i.e.*,

$$\delta_u \geq \sigma_I\big(S \cup \{u\}\big) - \sigma_I(S).$$

Before searching for the first node (*i.e.* $S = \emptyset$), we estimate an upper bound for each node by Corollary 1. Then, the algorithm proceeds similar to CELF. Note that by the properties of submodular, these upper bounds of marginal increments can be dynamically adjusted by MC simulations, which becomes smaller with the algorithm carrying on.

In the algorithm, $MC(S)$ denotes that we employ the Monte-Carlo simulation to estimate $\sigma_I(S)$ for the initial set S, $I(v) = 0$ denotes that the Monte-Carlo has not been used on the node v yet in the current iteration, $I(v) = 1$ means the Monte-Carlo simulation has already been computed on the node v.

C. Discussions on the upper bound

We have derived the upper bound for the spread function $\sigma_I(S)$, and developed a new UBLF algorithm. One may have the following concern: *How large is the gap between the estimated upper bound and the real value of $\sigma_I(S)$?*

In this part, we aim to explain that, under Conditions **(I)** the propagation probability $\{pp(u,v)\}$ is relatively small, and **(II)** the number of nodes N is large enough, the upper bound asymptotically approximates the real value of $\sigma_I(S)$.

Formally, if the two conditions are met, we can relax Eq.(15) to Eq.(17) as follow,

$$\sigma_I(S) \approx \Pi_0^S \cdot (E - PP)^{-1} \cdot \mathbf{1} \tag{17}$$

In the sequel, we will explain why Eq.(17) holds under the two given conditions. We first present two lemmas, based on which we derive the result in Eq.(17).

Lemma 1: For small positive numbers $x_1, x_2, \ldots, x_n$, it follows that

$$1 - \prod_{i=1}^{n}(1 - x_i) \approx \sum_{i=1}^{n} x_i. \tag{18}$$

Proof: Note that

$$\prod_{i=1}^{n}(1 - x_i) = 1 - \sum_{i=1}^{n} x_i + o(\sum_{1 \leq i < j \leq n} x_i x_j)$$

when $x_1, \ldots, x_n$ are relatively small, we obtain Eq.(18). □

We use Example 3 to explain Lemma 1.

Example 3: In Fig.2, nodes w and u are newly activated at step t, and they are both parents of v, with $pp(w,v) = 0.1$ and $pp(u,v) = 0.2$. Then, the probability of node v being activated at step $t+1$ under the IC model is

$$1 - (1 - 0.1)(1 - 0.2) = 0.28,$$

and we can observe that the probability is also roughly equivalent to the value

$$0.1 + 0.2 = 0.3.$$

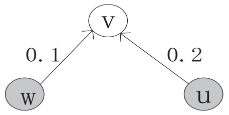


Figure 2. An example of 1D intervals

The two values are closer if the propagation probabilities $pp(w,v)$ and $pp(u,v)$ become smaller. □

Based on Lemma 1, we have

$$1 - \prod_{u \in S_{t-1}} (1 - pp(u,v)) \approx \sum_{u \in S_{t-1}} pp(u,v) \qquad (19)$$

Lemma 2: If the number of node N is large enough, we have

$$\mathbb{P}^S\big(v \notin \cup_{r=0}^{t-1} S_r \big| u \in S_{t-1}\big) \approx 1 \qquad (20)$$

Proof: If the node number N is very large, there are only few nodes in the set $\cup_{r=0}^{t-1} S_r$, and most nodes are not activated. □

We incorporate the two lemmas into the proof of Proposition 2, and obtain a relaxed version of Eq.(5) as follow,

$$\mathbb{P}^S(v \in S_t) \approx \sum_{u \in V} \mathbb{P}^S\big(u \in S_{t-1}\big) pp(u,v).$$

By using the matrix form, we rewrite the above approximation as follows,

$$\Pi_t^S \approx \Pi_{t-1}^S \cdot PP \qquad (21)$$

Incorporating the above approximation into the proofs of Theorem 1 and Corollary 1, we obtain the final result in Eq.(17).

To sum up, when the two given conditions are satisfied, the upper bound well approximates the spread function $\sigma_I(S)$. Hence, we have high accuracy guarantee to use the bound, $(E-PP)^{-1} \cdot \mathbf{1}$, as the pruning criterion. Specifically, we can choose k nodes with the highest values in the column vector $(E-PP)^{-1} \cdot \mathbf{1}$ as the initial seed set. For instance, in Example 1, we have

$$(E - PP)^{-1} \cdot \mathbf{1} = \begin{pmatrix} 1.3965 \\ 1.3684 \\ 1.2279 \\ 1.1396 \end{pmatrix}$$

If $k = 1$, we can simply choose node ① as the most influential seed node. If $k = 2$, we choose nodes ① and ② as the most influential seed set. We call this approach as the **Upper Bound based algorithm** (UBound in short). The algorithm is summarized below.

Algorithm 3: UBound

```
Input: the propagation probability matrix $PP$ of a
        graph $G = (V,E)$, a budget $k$
Output: The most influential set $S$ with $k$ nodes
Score ← $(E - PP)^{-1} \cdot \mathbf{1}$
Select the biggest $k$ nodes in Score as the output $S$
```

IV. Experiments

We conduct experiments on four real-world data sets to evaluate the upper bound and the **UBLF** algorithm. We implement the algorithms in C++ with the Standard Template Library (STL). All experiments are run on a Linux (RedHat 4) machine with a 2.33GHz Intel Xeon CPU and 16GB memory.

A. Data Sets

We use four real-world data sets for testing and comparisons. The data sets ca-GrQc[1] and ca-HepPh[2] are obtained from the e-print arXiv, which describe scientific collaborations between authors. If an author A co-authors a paper with author B, the graph contains a undirected edge from A to B. If a paper is co-authored by k authors, it generates a completely connected (sub)graph on k nodes. The third data set, Digger[3], is about stories promoted to Digg's front page, depicting the explicit friendships (or contacts) among users. The fourth data set, email-Enron[4], was originally made public by the Federal Energy Regulatory Commission during its investigation. Nodes of the network are email addresses, and if an address C sent one or multiple emails to address D, the graph contains an edge from C to D. The details of the data sets are listed in Table II.

Table II
Statistics of the four real-world networks.

Dataset	ca-GrQc	Digger	ca-HepPh	email-Enron
#Node	5,242	8,193	12,008	36,692
#Edge	28,980	56,440	237,010	367,662
Average Degree	5.5	6.9	19.7	10.0
Maximal Degree	320	850	1,346	7,792

B. Benchmark methods

We implement other five algorithms, CELF [16], DEGREE [14], PAGERANK [3], DEGREEDISCOUNT [4] and RANDOM, for comparisons.

- CELF [16]. We take $R = 10000$ in the Monte-Carlo simulation to estimate the spread of a seed set in CELF.
- DEGREE[14]. A heuristic algorithm based on the notion of "degree centrality", with high-degree nodes be

[1] http://snap.stanford.edu/data/
[2] http://snap.stanford.edu/data/
[3] http://arnetminer.org/heterinf
[4] http://snap.stanford.edu/data/

Table III
NUMBERS OF MONTE-CARLO SIMULATIONS IN THE FIRST 10 ITERATIONS

Datasets	Algorihms	1	2	3	4	5	6	7	8	9	10	Sum
ca-GrQc	CELF	**5,242**	4158	14	12	10	3	15	1	3	1	**9459**
	UBLF	**86**	44	10	6	20	13	11	80	70	76	**416**
Digger	CELF	**8,193**	8192	6	7	42	33	272	233	836	475	**18,289**
	UBLF	**245**	10	3	5	27	9	8	2	15	35	**359**
ca-HepPh	CELF	**12,008**	11204	13	3	19	3	5	7	21	1	**23,284**
	UBLF	**260**	28	22	135	36	8	3	22	69	24	**607**
email-Enron	CELF	**36,692**	33696	20	16	14	7	4	28	5	6	**70,488**
	UBLF	**19**	13	8	8	21	20	5	13	29	31	**167**

used as influential ones. The seeds are the nodes with the k highest out-degrees.

- PAGERANK [3]. A link analysis algorithm which ranks the importance of pages in a Web graph. We implement the power method with a damping factor of 0.85 and pick the k highest-ranked nodes as seeds. The algorithm stops if the score vectors from two consecutive iterations differ by at most 10^{-6} with the L_1-norm.
- DEGREEDISCOUNT [4]. A degree discount heuristic algorithm developed for the IC model with uniform propagation probability.
- RANDOM. It simply selects k random vertices in the graph as the seed set, which is taken as the baseline.

The simple greedy algorithm are not reported in this work, as there have been plenty of work reporting that CELF has the same influence spread result and less running time than the simple greedy algorithm. We also ignore the CELF++ [10] algorithm, as its performance has almost the same magnitude as CELF, with only $35\% - 55\%$ faster than CELF. Furthermore, we ignore other centrality measures, such as the distance centrality and betweenness centrality as heuristics, as it has been shown in [4] that distance centrality is very slow and has very poor influence spread, while betweenness centrality would be much slower than distance centrality.

C. Experimental Results

In our experiments, to obtain the influence spread of the heuristic algorithms for each seed set, we run the simulation on the networks $10,000$ times and take the average of the influence spread, which matches the accuracy of the greedy algorithms.

We assign a uniform propagation probability of $p = 0.01$ to each directed link in the network for the IC model, *i.e.*, $pp(u, v) = p$ for any directed edge $(u, v) \in E$. One should note that an undirected graphs can also be regarded as a directed graph by treating an undirected link as bidirectional.

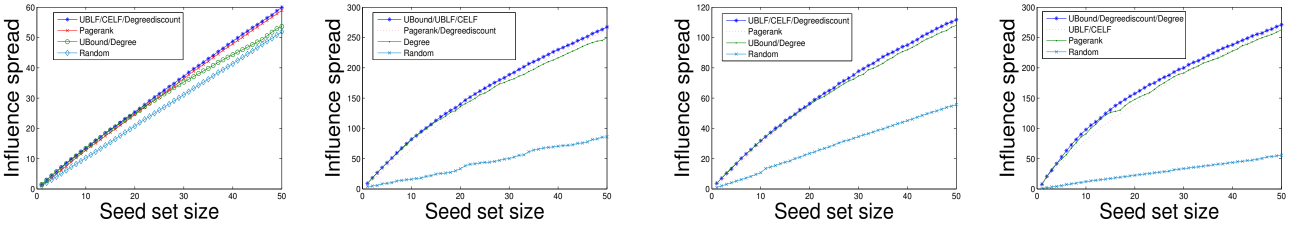
Number of Monte-Carlo calls. We compare the number of Monte-Carlo calls between CELF and UBLF. In Table III, we record the number of Monte-Carlo calls in the first 10 iterations on the four data sets. From the results, we can observe that the number of Monte-Carlo calls in UBLF is significantly reduced, compared to that in CELF, especially in the first two iterations.

The reason is because CELF has to call N estimation procedures in the first iteration to establish the initial upper bounds of marginal increments. In contrast, in UBLF, we use the upper bound to estimate the spread, instead of actually calculating Monte-Carlos. Therefore, UBLF can significantly reduce the number of Monte-Carlo calls, especially in the first iteration.

One may notice that in the last several steps in Table III, CELF occasionally defeats our method. This is because the upper bound given by Eq. (15) has strict constraints, given in Eq.(14). During the algorithm running time, these constraints may not always be satisfied. Thus, we can observe that CELF sometimes defeats our method. For example, at the last step on the first data set, CELF has 1 MC call but UBLF has 76 calls. Fortunately, the total call number of UBLF is much less than CELF. As listed in the last column, the total call number of the first 10 iterations of UBLF, compared to CELF, is reduced at a rate of 95.6%, 98.0%, 97.4%, 99.8% on the four data sets, respectively.

Influence spread. Influence spread is an important measure for comparisons. In this part, we run tests on the four data sets to obtain influence spread results *w.r.t.* parameter k (the seed set size), where k increases from 1 to 50. We list the results in Fig. 3. For easy reading, in the figure, the legend ranks the algorithms top-down based on their average influence spread values. If two curves are too close, we group them together in the legend.

From the results, we can observe that UBLF, as an updated version of CELF and the simple greedy algorithm, has competitive spread results in the four data sets. Our heuristic algorithm, Ubound, also performs well on average. An important observation is that the spreads of UBLF and CELF are completely identical in the four figures, which explains again that UBLF and CELF share the same logic in selecting nodes, the only difference is the number of Monte-Carlo calls. One may notice that the spreads in the data set ca-GrQc have small ranges. This is because the network is



(a) ca-GrQc

(b) Digger

(c) ca-HepPh

(d) email-Enron

Figure 3. Influence spread results *w.r.t.* seed size k on the four data sets.

built by cooperations and the edges are sparse and relative.

Time cost. Fig. 4 shows time costs of selecting 10 seeds. From the results, we can observe that the heuristic algorithms, Degree, DegreeDiscount and PageRank, are very fast in selecting candidate nodes (it normally takes only 1 second or less). The Ubound algorithm is slightly slower than the above three algorithms. Considering UBLF reduces 95% Monte-Carlo calls (as shown in Table III), the runtime here is acceptable. From Fig. 4, we can observe that UBLF is **2-5** times faster than CELF. One may question that such a low improvement can be neglected in large networks. In fact, UBLF scales well to large networks. Note that with the networks increase, Monte-Carlo simulations take heavier time, and UBLF will achieve better performance gain by pruning unnecessary Monte-Carlo simulations. In this experiment, because the network is not in huge-scale, the Monte-Carlo simulations are not very time-consuming, and the absolute time reduction is not significant. In the future, we will test and compare time costs on larger networks with high performance computing systems.

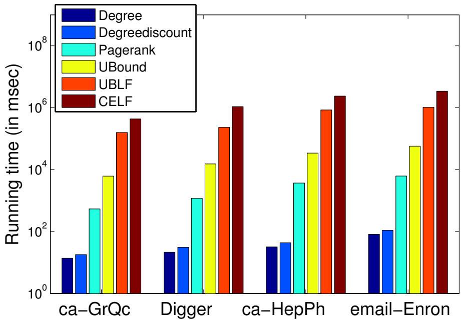


Figure 4. Runtimes for different algorithms.

V. Related work

Domingos and Richardson [7], [20] first formulated the influence maximization problem as an algorithmic problem in probabilistic methods. Later, Kempe et al. [14] first modeled the problem as the discrete optimization problem, as described in Section II. A major limitation of their approaches is computational inefficiency on large networks. Thus, considerable work have been conducted to find efficient solutions.

To address the **inefficiency of Monte-Carlo simulation** problem, a line of heuristic algorithms have been proposed. Chen et al. [4] proposed *degree discount heuristics* under the uniform IC model, in which all edge probabilities are the same. Experimental results indicate that the new heuristics are efficient with reasonably good influence spread guarantee. Following the same idea, Chen et al. [5] proposed a Maximum Influence Arborescence(MIA) model to evaluate the influence spread of a user. In [15], Kimura and Saito proposed *shortest-path* based influence cascade models and efficient algorithms to compute influence spread under modified IC models.

For the LT model, Chen et al. [6] observed that while computing the spread is $\#P$-hard in general graphs under the LT model, it can be computed in linear time on directed acyclic graphs (DAGs). Moreover, they claim that the majority of the influence flows only within a small neighborhood and thus they construct one local DAG (LDAG) per node in the graph and assume that the influence flows to the node only through that LDAG. They experimentally show that this heuristic is significantly faster than the greedy algorithm and achieves high quality seed set selection, measured in terms of the spread achieved. Similarly, Goyal et al. in [11] proposed SIMPATH, an efficient and effective algorithm for influence maximization under the LT model that addresses these drawbacks by incorporating several clever optimizations. In [18], Narayanam and Narahari proposed a Shapley value based heuristic SPIN for the LT model. However, SPIN only relies on the evaluation of influence spreads of seed sets, and thus does not use specific features of the LT model. Moreover, SPIN is not scalable, with running time comparable (as shown in [18]) or slower than the optimized greedy algorithm.

To overcome the **inefficiency of the simple greedy algorithm**, Lescovec et al. in [16] proposed the CELF algorithm, which makes 700 times improvement on the simple greedy algorithm. Following the same idea, Goyal et al. in [10], proposed CELF++, an extension to CELF that further reduces the number of spread estimation calls. The key idea behind CELF++ is that in any iteration, whenever the marginal gain of a node u is computed $w.r.t.$ the current seed set S, the algorithm also computes the marginal gain of u $w.r.t.$ $S \cup \{x\}$, where x is the node that has the maximum marginal gain among all the nodes examined in the current iteration until now. Thus, if x is chosen as the seed node at the end of the current iteration, there is no need to recompute the marginal gain of u in the next iteration. Clearly, the algorithm performs well when it is possible to compute the marginal gain of a node u *w.r.t.* S and $S \cup \{x\}$ simultaneously without much overhead. The authors of [10] report that CELF++ is found to be approximately $35\%-55\%$ faster than CELF.

In addition to the above works, Wang et al. [23] discussed the influence maximization from the view of community. They proposed a new algorithm called Community based Greedy algorithm for mining top-K influential nodes. Barbieri et al. [1] studied social influence from a topic modeling perspective. Corresponding to the standard problem, Guo et al. [12] investigated the personal influence maximization problem. Goyal et al. [9] proposed an alternative approach to influence maximization which, instead of assuming influence probabilities are given as input, directly uses the past available data. The complementary problem of learning influence probabilities from the available data is studied in the works [8], [21] and [22].

VI. Conclusion

In this paper, we derived an upper bound for the spread function in solving influence maximization problem in social networks. Based on the bound, we proposed a new Upper Bound based Lazy Forward algorithm (**UBLF** in short). Compared with CELF, UBLF significantly reduces the number of Monte-Carlo calls, *e.g.*, more than **95% reduction of Monte-Carlo calls** than CELF in our experiments. The experimental results also verify that UBLF can enhance CELF's efficiency by **2-5 times** at least when the seed set is small.

There are several interesting future directions. First, the upper bound in this study was derived under the IC model, its validation under the LT model or other influence propagation models is unknown and needs to be further studied. Second, we mainly targeted the CELF algorithm, and used the upper bound to improve CELF, leading to a new UBLF algorithm. Incorporating the upper bound into other heuristic algorithms is also an interesting research topic.

Acknowledgment

This work was supported by the NSFC (No. 61003167), IIE Chinese Academy of Sciences (No. Y3Z0062101), 863 projects (No. 2011AA010703 and 2012AA012502), 973 project (No. 2013CB329606), and the Strategic Leading Science and Technology Projects of Chinese Academy of Sciences (No.XDA06030200).

References

[1] N. Barbieri, F. Bonchi, and G. Manco, ”Topic-aware Social Influence Propagation Models,” in ICDM 2012.

[2] F. Bonchi, ”Influence propagation in social networks: A data mining perspective,” IEEE Intelligent Informatics Bulletin, Vol.12, No.1, 2011.

[3] S. Brin, and L. Page, ”The anatomy of a large-scale hypertextual web search engine,” Computer Networks, vol. 30, no. 1-7, pp. 107-117, 1998.

[4] W. Chen, Y. Wang, and S. Yang, ”Efficient influence maximization in social networks,” in KDD 2009.

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

[6] W. Chen, Y. Yuan, and L. Zhang, ”Scalable influence maximization in social networks under the linear threshold model,” in ICDM 2010.

[7] P. Domingos and M. Richardson, ”Mining the network value of customers,” in KDD 2001.

[8] A. Goyal, F. Bonchi, and L. V. Lakshmanan, ”Learning influence probabilities in social networks,” in WSDM 2010.

[9] A. Goyal, F. Bonchi, and L. V. Lakshmanan, ”A data-based approach to social influence maximization,” in PVLDB 2012.

[10] A. Goyal, W. Lu, and L. V. S. Lakshmanan, ”CELF++: Optimiz-ing the Greedy Algorithm for Influence Maximization in Social Networks,” in WWW 2011.

[11] A. Goyal, W. Lu, and L. V. S. Lakshmanan, ”SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model,” in ICDM 2011.

[12] J. Guo, P. Zhang, C. Zhou, Y. Cao, and L. Guo, ”Personalized influence maximization on social networks,” in CIKM 2013.

[13] R. A. Horn and C. R. Johnson, ”Matrix analysis,” Cambridge university press, 1990.

[14] D. Kempe, J. M. Kleinberg, and É. Tardos, ”Maximizing the spread of influence through a social network,” in KDD 2003.

[15] M. Kimura and K. Saito, ”Tractable models for information diffusion in social networks,” in ECML PKDD 2006.

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

[17] E. Mossel and S. Rich, ”On the Submodularity of Influence in Social Networks,” in STOC 2007.

[18] R. Narayanam and Y. Narahari, ”A shapley value based approach to discover influential nodes in social networks,” in TASAE 2010.

[19] G. Nemhauser, L. Wolsey, and M. Fisher, ”An analysis of the approximations for maximizing submodular set functions,” Mathematical Programming, 14: 265-294, 1978.

[20] M. Richardson and P. Domingos, ”Mining knowledge-sharing sites for viral marketing,” in KDD 2002.

[21] K. Saito, R. Nakano, and M. Kimura, ”Prediction of information diffusion probabilities for independent cascade model,” in KES 2008.

[22] J. Tang, J. Sun, C. Wang, and Z. Yang, ”Social influence analysis in large-scale networks,” in KDD 2009.

[23] Y. Wang, G. Cong, G. Song, and K. Xie, ”Community-based greedy algorithm for mining top-k influential nodes in mobile social networks,” in KDD 2010.