Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations

Naoto Ohsaka [#,†], **Takuya Akiba** [#,†], **Yuichi Yoshida** [*,§] and **Ken-ichi Kawarabayashi** [*,†]
[#]The University of Tokyo, Tokyo, Japan, [*]National Institute of Informatics, Tokyo, Japan
[§]Preferred Infrastructure, Inc., Tokyo, Japan, [†]JST, ERATO, Kawarabayashi Large Graph Project, Japan
{ohsaka, t.akiba}@is.s.u-tokyo.ac.jp, {yyoshida, k_keniti}@nii.ac.jp

Abstract

Influence maximization is a problem to find small sets of highly influential individuals in a social network to maximize the spread of influence under stochastic cascade models of propagation. Although the problem has been well-studied, it is still highly challenging to find solutions of high quality in large-scale networks of the day. While Monte-Carlo-simulation-based methods produce near-optimal solutions with a theoretical guarantee, they are prohibitively slow for large graphs. As a result, many heuristic methods without any theoretical guarantee have been developed, but all of them substantially compromise solution quality. To address this issue, we propose a new method for the influence maximization problem. Unlike other recent heuristic methods, the proposed method is a Monte-Carlo-simulation-based method, and thus it consistently produces solutions of high quality with the theoretical guarantee. On the other hand, unlike other previous Monte-Carlo-simulation-based methods, it runs as fast as other state-of-the-art methods, and can be applied to large networks of the day. Through our extensive experiments, we demonstrate the scalability and the solution quality of the proposed method.

Introduction

Viral marketing is a cost-effective marketing strategy that promotes products by giving free or discounted items to a selected group of highly influential individuals, in the hope that through the "word-of-mouth" effects, a large number of product adoptions will occur (Domingos and Richardson 2001; Richardson and Domingos 2002). Viral marketing is based on the power of word-of-mouth effect, i.e., the reputation came from friends or families by word-of-mouth communication are much more trusted than advertisement statements. Therefore, finding small but effective seed sets by analyzing social networks is the key for successful viral marketing, and thus has been intensively studied.

The problem to find such seed sets is called *influence maximization* and mathematically formalized under a stochastic cascade model of propagation (Kempe, Kleinberg, and Tardos 2003). This influence maximization problem is NP-hard, and thus many approximation algorithms and heuristics have been developed. However, all these previous methods still suffer from either low scalability or low precision.

One of the most classical approaches is based on Monte-Carlo-simulation-based greedy algorithms (Kempe, Kleinberg, and Tardos 2003). Due to a nice property of the problem called *submodularity*, the greedy algorithms produce near-optimal solutions with a theoretical guarantee. However, even with some speed-up techniques (Leskovec et al. 2007; Chen, Wang, and Yang 2009; Wang et al. 2010; Goyal, Lu, and Lakshmanan 2011; Cheng et al. 2013), they have not been applicable to large-scale networks with millions of vertices because of its high time complexity for simulating the influence cascade.

As a result, a plethora of heuristic methods without any theoretical guarantee have been developed. However, while they are more scalable than the greedy algorithms, some of them substantially compromise solution quality, and others that seemingly perform well on some cases often turned out not to be robust to network structures or parameter settings (as will be shown in our experiments).

To address this issue, we propose a new efficient algorithm for the influence maximization problem. Unlike recent heuristic methods, our method is a Monte-Carlo-simulation-based method, i.e., it estimates the spread of influence by reachability tests (Kempe, Kleinberg, and Tardos 2003). Hence the solution quality is theoretically guaranteed. Indeed, as we will see in the experimental results, the solution quality is almost always better than all the other heuristic methods. Nevertheless, while our algorithm is based on Monte-Carlo simulations, our new techniques drastically reduce the computational complexity and enable it to work on very large networks with tens of millions of edges in comparable time to previous heuristic methods.

The main ingredients for obtaining our scalable greedy algorithm are the following techniques: (1) we maintain and incrementally update the outcome of Monte-Carlo simulations to reduce the simulation cost and the number of necessary simulations, (2) we effectively prune breadth first searches (BFSs) for reachability tests in the simulation outcome, and (3) we detect and avoid the unnecessary recomputations of vertex scores. In particular, with regard to the point (1), we provide not only empirical results, but also theoretical evidence that the technique really reduces the number of

necessary simulations to accurately identify highly influential vertices.

In our experiments, we confirm that (1) the proposed method runs as fast as other state-of-the-art methods on large-scale networks of the day, and (2) the solution quality is consistently high and almost always better than all other heuristic methods.

Related Work

Monte-Carlo-Simulation-based Methods

Monte-Carlo-simulation-based methods produce highly influential seed sets because they can accurately estimate the influence spread. However, it is challenging to apply them to large-scale networks.

Kempe et al. (Kempe, Kleinberg, and Tardos 2003) is the first to propose a greedy algorithm for influence maximization. Due to a nice property called submodularity, the greedy algorithm produces near-optimal solutions with a theoretical guarantee. However, it suffers from scalability due to the large time complexity of Monte-Carlo simulations. The Cost-Effective Lazy Forward (CELF) (Leskovec et al. 2007) algorithm utilizes the submodularity for reducing the number of necessary influence estimations with the same performance as the original greedy algorithm. Although empirical results show 700 times speed-up, it still takes a few hours for graphs with tens of thousands of vertices. A sample average approximation approach which optimizes a fixed set of samples is proposed by (Sheldon et al. 2010). StaticGreedyDU (Cheng et al. 2013) is also based on sample average approximation. Reusing subgraphs generated from a given graph, it reduces the number of simulations by two orders of magnitude while keeping almost the same performance as the original greedy algorithm empirically. However, its scalability is still impractical for large graphs as will be shown in our experiments. In contrast, our method drastically reduces the simulation cost with novel pruning techniques. Recently, a near-linear time algorithm with a theoretical guarantee is proposed by (Borgs et al. 2014).

Heuristic-based Methods

To avoid using Monte-Carlo simulations, various heuristic-based methods have been proposed.

Simulated Annealing with Effective Diffusion Values (SAEDV) (Jiang et al. 2011), which is the first simulated-annealing-based approach, represents the influence spread as a simple heuristic equation. PMIA (Chen, Wang, and Wang 2010) adopts maximum influence paths to estimate the influence spread. The drawback is that the running time increases explosively as the probability of the spread increases (Jung, Heo, and Chen 2012). Influence Rank Influence Estimation (IRIE) (Jung, Heo, and Chen 2012) formulates the influence spread using simultaneous linear equations.

Heuristic-based methods are more scalable than Monte-Carlo-simulation-based methods, but the quality of solutions is not robust to network structures or parameter settings partly because of the absence of theoretical guarantees.

Preliminaries

Notations

Let $G = (V, E)$ be a directed graph with a vertex set V of size n and an edge set E of size m. The symbol $u \overset{G}{\rightsquigarrow} v$ means that a vertex v is reachable from a vertex u in the graph G, namely, there is a path from u to v in G.

Problem Definition

In this paper, we adopt the most standard information diffusion model called the *independent cascade* (IC) model (Kempe, Kleinberg, and Tardos 2003). In the IC model, given a directed graph $G = (V, E)$, a propagation probability function $p : E \to [0, 1]$, and a vertex set $S \subseteq V$ called a *seed* set, we first activate vertices in S. Then the process unfolds in discrete steps according to the following randomized rule. When a vertex u becomes active in the step t for the first time, it is given a single chance to activate each current inactive vertex v in the neighbors of u. It succeeds with probability p_{uv}. If u succeeds, then v will become active in the step $t + 1$. Whether or not u succeeds, it cannot make any further attempt to activate v in subsequent steps. The process runs until no more activation is possible. The *influence spread* of a seed set S under the IC model is defined as the expected total number of active vertices given a seed set S. We denote the influence spread of S by $\sigma(S)$. Formally, the *influence maximization problem under the IC model* is defined as the problem of finding a vertex set S of size k maximizing $\sigma(S)$, where k is a given parameter.

Greedy Algorithm with Constant Approximation

Though the influence maximization problem is proved to be NP-hard (Kempe, Kleinberg, and Tardos 2003), a natural greedy algorithm achieves a constant approximation to the optimum solution due to non-negativity, monotonicity and submodularity of $\sigma(\cdot)$. We say that a set function $f : 2^V \to \mathbb{R}$ is non-negative if $f(S) \geq 0$ for all $S \subseteq V$, and monotone if $f(S) \leq f(T)$ for all $S \subseteq T$, and submodular if $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ for all $S \subseteq T$ and $v \in V$. *The original greedy algorithm* (Kempe, Kleinberg, and Tardos 2003) starts with an empty seed set S, and then adds a vertex t with the maximum marginal influence, i.e., $t = \arg\max_{v \in V \setminus S} \sigma(S \cup \{v\}) - \sigma(S)$, into S until k vertices are added. The following theorems guarantee that the original greedy algorithm approximates the optimum solution within a factor of slightly better than 63% by evaluating the influence spread function nk times.

Theorem 1. *(Nemhauser, Wolsey, and Fisher 1978) For a non-negative, monotone submodular function f, let S be a set of size k obtained by the greedy strategy. Then, $f(S) \geq (1 - 1/e) f(S^*)$ where S^* is the optimum solution.*

Theorem 2. *(Kempe, Kleinberg, and Tardos 2003) For the IC model, the influence spread function $\sigma(\cdot)$ is non-negative, monotone and submodular.*

Computing Influence Spread

Since exact computation of $\sigma(\cdot)$ is #P-hard (Chen, Wang, and Wang 2010), Monte-Carlo simulations have been used

in previous works (Kempe, Kleinberg, and Tardos 2003; Leskovec et al. 2007; Chen, Wang, and Yang 2009; Wang et al. 2010; Cheng et al. 2013). The time complexity of Monte-Carlo simulations is $O(mR)$ where R is the number of simulations, and so the time complexity of the greedy algorithm is $O(kmnR)$. In addition, we should conduct a number of simulations (e.g., $R \approx 10,000$) to obtain a sufficiently accurate estimation of the influence spread function. To address this issue, we propose a fast influence maximization algorithm based on Monte-Carlo simulations in the next section.

Proposed Algorithm

We propose a fast and accurate algorithm for the influence maximization problem under the IC model. First, we describe the coin flip technique to demonstrate that simulating the IC model is equivalent to testing reachability on random graphs. Then, we provide an overview of our proposed algorithm's core. Finally, we introduce two speed-up techniques to make it faster and practical. Note that our proposed algorithm is based on the original greedy algorithm, and thus it gives near-optimal solutions with a theoretical guarantee.

Coin Flip Technique under the IC Model

In the "coin flip" technique (Kempe, Kleinberg, and Tardos 2003), let $\mathcal{D}_G(S)$ be the distribution of the vertex set influenced by S in the IC model, G_p be the random graph obtained from G by keeping each edge e with the probability p_e and $\mathcal{D}'_G(S)$ be the distribution of the vertex set reachable from S in G_p. Then the crucial insight is the following:

The two distributions $\mathcal{D}_G(S)$ and $\mathcal{D}'_G(S)$ are the same.

This fact tells us that we do not really have to consider the time of activations, but we just need to consider reachability on the random graph G_p.

Overview

We now explain our proposed algorithm. At a high level, our algorithm consists of the following ingredients: (1) generate random graphs from G, (2) construct vertex-weighted directed acyclic graphs (DAGs) from the random graphs, (3) approximate the value of marginal influence by averaging the total weight of vertices reachable from a single vertex in each DAG, (4) select a seed according to the greedy strategy, and (5) update DAGs. Note that our method reuses the outcome of coin flipping during the whole process, and so it is based on sample average approximation (Sheldon et al. 2010; Cheng et al. 2013).

A pseudocode of the proposed algorithm is shown in Algorithm 1. A parameter R indicates the number of DAGs. Our algorithm consists of two parts. First, we repeat the following process to generate R DAGs $G_1, G_2, \ldots, G_R$. In the i-th process, we first flip a coin for each edge, i.e., an edge $e \in E$ lives with probability p_e. Let E'_i be the edge set extracted from E in this manner. Then we compute strongly connected components (SCCs) of $G'_i = (V, E'_i)$ to obtain the following: a strongly connected component containing $v \in V$ (denoted by $\mathrm{comp}_i[v]$), the number of vertices in a strongly connected component v (denoted by $\mathrm{weight}_i[v]$). Consequently, the i-th vertex-weighted DAG $G_i = (V_i, E_i)$

Algorithm 1 Fast and accurate algorithm for influence maximization under the IC model

Require: $G = (V, E), p : E \to [0, 1], k, R$ (# of DAGs)
1: **for** $i = 1$ **to** R **do**
2: $E'_i \leftarrow$ edge set by keeping each edge e with prob. p_e
3: Compute SCCs of $G'_i = (V, E'_i)$
4: $G_i = (V_i, E_i) \leftarrow$ decomposed DAG of G'_i
5: $h_i \leftarrow$ a vertex with the maximum degree in V_i
6: $D_i \leftarrow \{v \in V_i \mid h_i \overset{G_i}{\rightsquigarrow} v\}$
7: $A_i \leftarrow \{v \in V_i \mid v \overset{G_i}{\rightsquigarrow} h_i\} \setminus \{h_i\}$
8: $\mathrm{latest}_i[v] \leftarrow$ `false` **for all** $v \in V_i$
9: $S \leftarrow \emptyset$
10: **while** $|S| < k$ **do**
11: $t \leftarrow \arg\max_{v \in V} \frac{1}{R} \sum_{i=1}^{R} \textsc{Gain}(i, v)$
12: $S \leftarrow S \cup \{t\}$
13: **for** $i = 1$ **to** R **do**
14: $\textsc{UpdateDAG}(i, t)$
15: **return** S

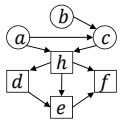
is constructed as follows: $V_i = \{\mathrm{comp}_i[v] \mid v \in V\}, E_i = \{(\mathrm{comp}_i[u], \mathrm{comp}_i[v]) \mid uv \in E'_i\}$. It is worth noting that the number of vertices reachable from a vertex set S ($S \subseteq V$) in the i-th random graph G'_i is equal to the total weight of vertices reachable from one of $\{\mathrm{comp}_i[t] \mid t \in S\}$ in the i-th vertex-weighted DAG G_i. We use the symbol $\sigma_{G_i}(S)$ to denote this value, that is,

$$\begin{aligned} \sigma_{G_i}(S) &= |\{v \in V \mid \exists t \in S, t \overset{G'_i}{\rightsquigarrow} v\}| \\ &= \sum_{v \in V_i : \exists t \in S, \mathrm{comp}_i[t] \overset{G_i}{\rightsquigarrow} v} \mathrm{weight}_i[v]. \end{aligned}$$

In addition, the value of $\sigma_{G_i}(S \cup \{v\}) - \sigma_{G_i}(S)$ is called the *gain* of v with regard to S in G_i and denoted by $\sigma_{G_i}(v \mid S)$. In the end of the i-th process, we select a hub vertex h_i with the maximum degree and compute descendants of h_i (denoted by D_i) and ancestors of h_i (denoted by A_i) for the first speed-up technique, and then initialize a flag $\mathrm{latest}_i[v]$ of gain recomputation for the second speed-up technique.

Second, we select a seed set according the greedy strategy, namely, our algorithm starts with an empty seed set $S = \emptyset$, and then adds a vertex t with the maximum marginal influence to S until k vertices are added. We call the phase of choosing the i-th seed vertex the *i-th phase*. The function $\textsc{Gain}(i, v)$ conducts a BFS from $\mathrm{comp}_i[v]$ to return the gain of v in G_i. An approximate value of marginal influence $\sigma(S \cup \{v\}) - \sigma(S)$ is obtained by averaging the gain of v in each DAG. After choosing a seed vertex t with the maximum average gain, the function $\textsc{UpdateDAG}(i, t)$ removes vertices reachable from $\mathrm{comp}_i[t]$ in G_i. Therefore the total weight of vertices reachable from $\mathrm{comp}_i[v]$ in G_i will be equal to $\sigma_{G_i}(v \mid S \cup \{t\})$.

Unfortunately, the present method does not substantially improve the time complexity of the original greedy algorithm because BFSs still need to be conducted approximately knR times. In order to make our algorithm faster and practical, we introduce two speed-up techniques. The first is *pruned BFS* which reduces the number of vertices visited during BFSs drastically. The second is the technique for *avoiding gain recomputations*. These techniques do not



Vertex	# of vertices visited during normal BFS	pruned BFS
a	6	2
b	6	2
c	5	1

Figure 1: An example of pruned BFS. Square vertices are temporarily removed during a BFS from a circular vertex.

Algorithm 2 Technique 1: Pruned BFS

```
1: function GAIN(i, v_V ∈ V)
2:     v ← comp_i[v_V]
3:     return 0 if v ∉ V_i
4:     return δ_i[v] if latest_i[v]
5:     latest_i[v] ← true
6:     if v ∈ A_i ∧ |S| = 0 then
7:         δ_i[v] ← GAIN(i, h_V) s.t. comp_i[h_V] = h_i
8:     else
9:         δ_i[v] ← 0
10:    Q ← a queue with only one element v
11:    X ← {v}
12:    while Q ≠ ∅ do
13:        Dequeue u from Q
14:        if v ∈ A_i ∧ u ∈ D_i ∧ |S| = 0 then
15:            continue
16:        δ_i[v] ← δ_i[v] + weight_i[u]
17:        for all uw ∈ E_i do
18:            if w ∉ X ∧ w ∈ V_i then
19:                Enqueue w onto Q
20:                X ← X ∪ {w}
21:    return δ_i[v]
```

affect the estimation of influence spread, and thus our algorithm has still performance guarantee.

Technique 1: Pruned BFS

We discuss how to reduce the number of vertices visited during BFSs. Let us consider the total running time of BFSs from every vertex. The total number of vertices visited during the whole BFS is at least (# of ancestors of h) $\times$ (# of descendants of h) for any vertex h. This can be quadratic in the number of vertices; hence the whole BFS may be too expensive. The key observation for reducing the running time of BFSs is the following:

> If a vertex v can reach to a vertex h, vertices reachable from h can be pruned during the BFS from v.

To explain pruned BFS, let us take Figure 1 for example. A vertex h in this figure is considered to be a hub because its degree is the maximum. The ancestors and descendants of h are $\{a, b, c\}$ and $\{h, d, e, f\}$, respectively. Starting a normal BFS from a, we will visit $\{a, c, h, d, e, f\}$. That is redundant because we already know that a can reach to h, so that it can also reach to descendants of h. Therefore we only visit two vertices $\{a, c\}$ by pruning $\{h, d, e, f\}$. Similarly, we visit $\{b, c\}$ during a pruned BFS from b, while we visit $\{b, c, h, d, e, f\}$ during a normal BFS from b.

Pruned BFS is described as Algorithm 2. Before starting a BFS from a vertex v, we check whether v is an ancestor of the hub vertex h_i or not. If so, we conduct a pruned BFS, that is, vertices reachable from h_i are temporarily removed. Otherwise, we conduct a normal BFS. Notice that

Algorithm 3 Technique 2: Avoiding gain recomputations

```
1: function UPDATEDAG(i, t_V ∈ V)
2:     t ← comp_i[t_V]
3:     for all v ∈ V_i : ∃u, t ⇝^{G_i} u ∧ v ⇝^{G_i} u do
4:         latest_i[v] ← false
5:     V_i ← V_i \ {v | t ⇝^{G_i} v}
```

the gain of v in G_i is exactly equal to the sum of the gain of h_i and the total weight of vertices visited during the pruned BFS. Pruned BFSs will be only conducted in the first phase because the next proposed speed-up technique reduces the number of gain recomputations after the first phase.

Technique 2: Avoiding Gain Recomputations

We discuss how to detect and avoid the unnecessary gain recomputations. The key observation to detect the unnecessary recomputations is that after adding a new seed t to a seed set, the gain of a vertex v in a DAG will change if and only if *one of the vertices reachable from v is also reachable from t in the DAG*, because descendants of t will be removed.

The technique for avoiding gain recomputations is described as Algorithm 3. We conduct a BFS from t, then a reverse BFS from the vertices reachable from t. A flag of gain recomputation is set to each vertex visited during the reverse BFS, i.e., $\mathrm{latest}_i[v]$ is set to false. Finally, descendants of t are removed from V_i.

Theoretical Guarantee on the Number of Simulations

In this section, we theoretically show that the reusing technique reduces the number of necessary simulations by the factor of $1/k$ compared to the original greedy algorithm. Specifically, the number of necessary simulations is $O(\frac{\log k + \log n}{\epsilon^2} \log \frac{1}{\delta})$ for some parameters ϵ and δ.

We can think of two strategies of sampling graphs when choosing the seed set. The first one is sampling a set of graphs at once and reusing it throughout phases, and the second one is resampling a set of graphs for each phase. We call the former one the *reusing algorithm* and the latter one the *resampling algorithm*.

We first introduce Hoeffding's inequality.

Theorem 3 (Hoeffding's inequality). *Let $X_1, \ldots, X_n$ be independent random variables in $[0, 1]$. Let $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. Then, we have $\Pr[|\overline{X} - X| > t] \leq 2\exp(-2nt^2)$.*

Lemma 4. *Let G be a graph and $\mathcal{S}$ be a family of vertex sets. Let $R = O(\frac{1}{\epsilon^2}\log|\mathcal{S}|\log\frac{1}{\delta})$ and $G_1, \ldots, G_R$ be a set of graphs sampled from G. Then, with probability at least $1 - \delta$, we have $\overline{\sigma}(S) = \sigma_G(S) \pm \epsilon n$ for every set $S \in \mathcal{S}$, where $\overline{\sigma}(S) = \frac{1}{R}\sum_{i=1}^{R} \sigma_{G_i}(S)$.*

Proof. Recall that $\sigma_{G_i}(S) \in [0, n]$. For a fixed set $S \in \mathcal{S}$, by applying Hoeffding's inequality on $\frac{1}{n}\sigma_{G_i}(S), \ldots, \frac{1}{n}\sigma_{G_R}(S)$, we have $\Pr[|\overline{\sigma}(S) - \sigma(S)| > \epsilon n] \leq 2\exp(-2\epsilon^2 R)$. By the union bound over all sets

Table 1: Datasets

Dataset	$\|V\|$	$\|E\|$	direction
Epinions	76K	509K	directed
DBLP	655K	2.0M	undirected
LiveJournal	4.8M	69M	directed
Twitter30days	6.2M	62M	directed

in $\mathcal{S}$, the probability that $|\overline{\sigma}(S) - \sigma(S)| \leq \epsilon n$ for every $S \in \mathcal{S}$ is at least $1 - 2\exp(-2\epsilon^2 R)|\mathcal{S}|$. By choosing $R = O(\frac{1}{\epsilon^2}\log|\mathcal{S}|\log\frac{1}{\delta})$, we have the desired bound. □

We now analyze the number of samples we need in the reusing algorithm and the resampling algorithm. We introduce definitions that are common to both algorithms.

Let G be a graph of n vertices. Fix k and for each $i \in \{1,\ldots,k\}$, let S_i be the seed vertex set chosen by the algorithm right after the i-th phase. We define $S_0 = \emptyset$ and S_k is the output of the algorithm. Also, for $i \in \{0,\ldots,k-1\}$, we define $\mathcal{S}_i$ as the family of vertex sets for which the algorithm calculates σ in the i-th phase. Namely, $\mathcal{S}_{i+1} = \{S_i \cup \{v\} \mid v\}$. Let $\mathcal{S} = \cup_{i=0}^{k-1}\mathcal{S}_i$. We note that $\mathcal{S}_i$ $(i = 1,\ldots,k-1)$ and hence $\mathcal{S}$ are random variables.

The reusing algorithm first samples graphs $G_1,\ldots,G_R$, and for each vertex set S, it estimates $\sigma_G(S)$ by $\overline{\sigma}(S) := \frac{1}{R}\sum_{i=1}^{R}\sigma_{G_i}(S)$. We have the following.

Theorem 5. *By setting $R = O(\frac{\log k+\log n}{\epsilon^2}\log\frac{1}{\delta})$, with probability at least $1-\delta$, the estimates of the reusing algorithm satisfy that $\overline{\sigma}(S) = \sigma_G(S) \pm \epsilon n$ for every vertex set $S \in \mathcal{S}$.*

Proof. Note that $|\mathcal{S}| \leq kn$. Hence, the theorem is a direct consequence of Lemma 4. □

The resampling algorithm works as follows. In the i-th phase, it samples graphs $G_1^i,\ldots,G_R^i$, and for each vertex set S, it estimates $\sigma_G(S)$ by $\overline{\sigma}^i(S) := \frac{1}{R}\sum_{j=1}^{R}\sigma_{G_j^i}(S)$.

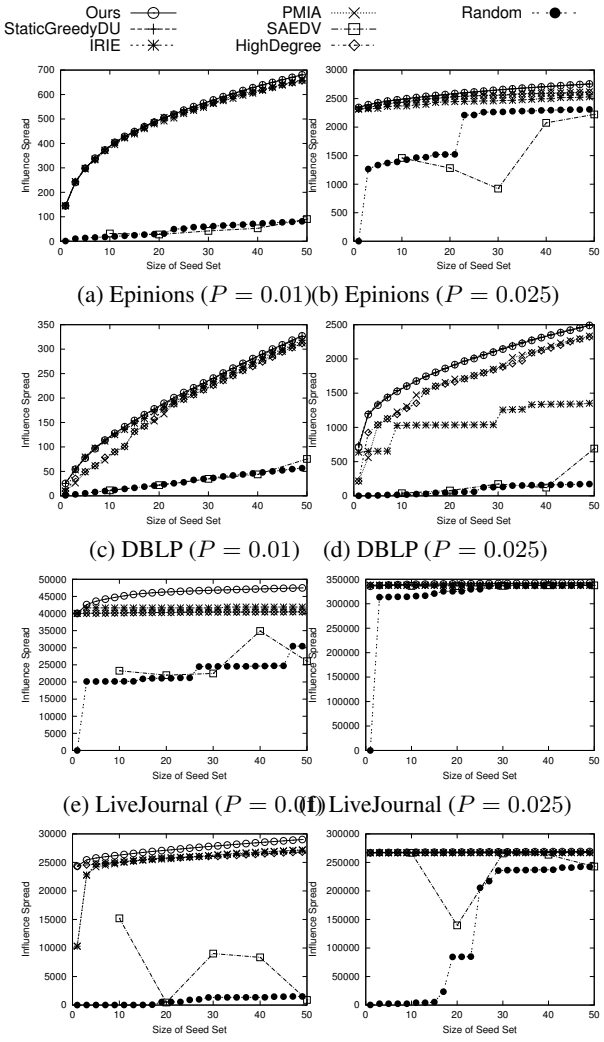
Theorem 6. *By setting $R = O(\frac{\log n}{\epsilon^2}\log\frac{1}{\delta})$, with probability at least $1-\delta$, the estimates of the resampling algorithm satisfy that $\overline{\sigma}^i(S) = \sigma_G(S) \pm \epsilon n$ for every i and every vertex set $S \in \mathcal{S}_i$.*

Proof. Note that $|\mathcal{S}_i| \leq n$. Hence, the theorem is a direct consequence of Lemma 4. □

Note that the reusing algorithm samples $O(\frac{\log k+\log n}{\epsilon^2}\log\frac{1}{\delta})$ times in total whereas the resampling algorithm samples $O(\frac{k\log n}{\epsilon^2}\log\frac{1}{\delta})$ times in total. Hence in view of the number of samples, the reusing algorithm is much efficient.

Experiments

We conducted experiments on several social networks to demonstrate efficiency and robustness of our algorithm by comparing with state-of-the-art existing algorithms.



(a) Epinions ($P = 0.01$) (b) Epinions ($P = 0.025$)

(c) DBLP ($P = 0.01$) (d) DBLP ($P = 0.025$)

(e) LiveJournal ($P = 0.01$) (f) LiveJournal ($P = 0.025$)

(g) Twitter30days ($P = 0.01$) (h) Twitter30days ($P = 0.025$)

Figure 2: Comparisons of influence spreads between the proposed algorithm and previous algorithms. Our algorithm is almost always the best among state-of-the-art algorithms.

Experimental Setup

Datasets We use four social networks: three directed graphs and one undirected graph (treated as a bidirected graph). The number of vertices and edges and the type of direction for each graph are summarized in Table 1. Detailed description of datasets is as follows.

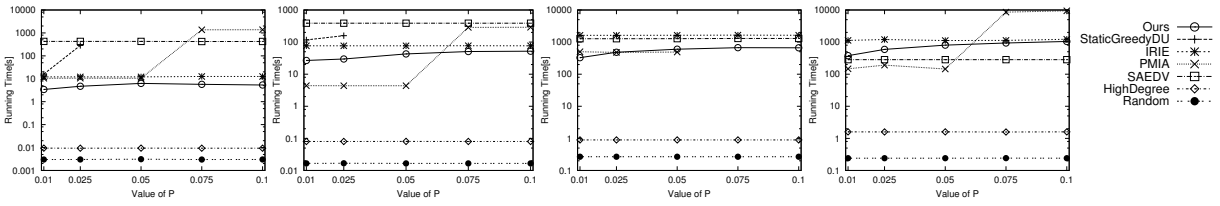
Epinions[1]**:** This is a who-trust-whom network of Epinions.com (www.epinions.com) where each vertex represents a user and each edge represents a trust relationship.

DBLP[2]**:** This is a collaboration network obtained from the DBLP Computer Science Bibliography Database.

LiveJournal[1]**:** This is an on-line social network in LiveJournal (www.livejournal.com) where each vertex represents a user and each edge represents a trust relationship.

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

[2] http://research.microsoft.com/en-us/people/weic/



(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter30days

Figure 3: Comparisons of running times to compute a seed set of size 50 between the proposed algorithm and previous algorithms. Our algorithm is comparable to IRIE and SAEDV, and more robust compared to PMIA and StaticGreedyDU.

Twitter30days: This is a graph obtained from Japanese tweets extracted from Twitter (twitter.com) in April 2013 where each vertex represents a user and two vertices are connected if one user replied to the other user.

Propagation Probabilities The probability of each edge is set to a parameter P. In order to investigate the behavior of each algorithm below, with some probability of influence, the value of P is set to $P = 0.01, 0.025, 0.05, 0.075, 0.1$.

Algorithms We compare our algorithm with the following four state-of-the-art algorithms and two baseline algorithms.
Ours: Our proposed algorithm for the IC model. The number of DAGs R is set to 200.
StaticGreedyDU (Cheng et al. 2013): StaticGreedyDU reuses the generated subgraphs to reduce the number of simulations. The number of subgraphs R is set to 200.
IRIE (Jung, Heo, and Chen 2012): IRIE represents the influence spread as simultaneous linear equations. The values of α and θ are set to 0.7 and $1/320$, respectively.
PMIA (Chen, Wang, and Wang 2010): PMIA adopts maximum influence paths for influence spread estimation. The value of a parameter θ is set to $1/320$.
SAEDV (Jiang et al. 2011): SAEDV is a simulated-annealing-based method. The values of parameters are set to the same as (Jiang et al. 2011).
HighDegree: Select k vertices in decreasing degree order.
Random: Select k vertices from a vertex set randomly.

Environments We conducted experiments on a Linux server with Intel Xeon X5670 (2.93 GHz) and 48GB for main memory. All algorithms were implemented in C++.

Results

Influence spread Figure 2 shows influence spreads on various graphs with respect to the size k of a seed set for each algorithm. The value of k is set from 1 to 50. We set $k = 10, 20, 30, 40, 50$ for SAEDV because it is not based on the greedy strategy. PMIA crashed due to out-of-memory on LiveJournal ($P \geq 0.075$). StaticGreedyDU crashed on Epinions ($P \geq 0.05$), DBLP ($P \geq 0.05$), LiveJournal ($P \geq 0.01$), and Twitter30days ($P \geq 0.01$). Thus we were unable to obtain seed sets in these settings. We ran Monte-Carlo simulations $10,000$ times and took the average in order to obtain reasonable estimates of the influence spread. We omit results for $P = 0.05, 0.075, 0.1$ because all state-of-the-art algorithms showed a similar behavior.

Ours is almost the best in all settings. StaticGreedyDU is also nearly the best. All heuristic-based methods showed approximately 13% and 7% less influence spread than Ours on LiveJournal ($P = 0.01$) and Twitter30days ($P = 0.01$), respectively. In particular, the influence spread of the seed set of size 50 given by IRIE (1,354) is approximately half of that given by Ours (2,510) on DBLP ($P = 0.025$). Similarly, the influence spread of the seed set of size 1 given by PMIA (216) is less than one-third of that given by Ours (713) on DBLP ($P = 0.025$). SAEDV gives seed sets of low quality in all settings. As can be seen, Random and HighDegree are not smart strategies.

Running time Figure 3 shows the running time for computing a seed set of size 50 with the value of P for each algorithm. Note that each running time does not include the time for reading the input graph from a secondary storage. We were unable to obtain results in the settings mentioned above due to out-of-memory. The fastest method is Random and the second is HighDegree. StaticGreedyDU did not work in many settings due to out-of-memory. Actually, it takes longer than one hour before the out-of-memory error occurs. The running time of PMIA increases explosively when the value of P exceeds 0.075. The performances of IRIE and SAEDV are not affected by the value of P. Although Ours is not always the fastest among other state-of-the-art algorithms, its running time is comparable to IRIE and SAEDV, and more robust compared to StaticGreedyDU and PMIA in terms of the value of P.

From these results, we can conclude that our algorithm is more robust than state-of-the-art algorithms to network structures or parameter settings, as well as it almost always gives the best solutions among state-of-the-art algorithms.

Conclusion

In this paper, we proposed a fast and accurate algorithm for the influence maximization problem under the independent cascade model. The proposed method exploits the existence of a hub in social networks to accelerate breadth first searches for reachability tests without loss of solution quality. In addition, we provided a theoretical guarantee that our method reduces the number of necessary simulations to select a seed set accurately. Our experimental results demonstrated that our method works on large networks with tens of millions of edges in comparable time to previous heuristic methods whereas it almost always produces the best solution among state-of-the-art methods in various settings.

Acknowledgment

We would like to thank to Kyomin Jung for providing us the codes of IRIE and PMIA, Guojie Song for providing us the code of SAEDV, and Suqi Cheng for providing us the code of StaticGreedyDU.

Takuya Akiba is supported by Grant-in-Aid for JSPS Fellows (256563). Yuichi Yoshida is supported by JSPS Grant-in-Aid for Research Activity Start-up (24800082), MEXT Grant-in-Aid for Scientific Research on Innovative Areas (24106003), and JST, ERATO, Kawarabayashi Large Graph Project.

References

Borgs, C.; Brautbar, M.; Chayes, J.; and Lucier, B. 2014. Maximizing Social Influence in Nearly Optimal Time. In *SODA*, 946–957.

Chen, W.; Wang, C.; and Wang, Y. 2010. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *KDD*, 1029–1038.

Chen, W.; Wang, Y.; and Yang, S. 2009. Efficient Influence Maximization in Social Networks. In *KDD*, 199–208.

Cheng, S.; Shen, H.; Huang, J.; Zhang, G.; and Cheng, X. 2013. StaticGreedy: Solving the Scalability-Accuracy Dilemma in Influence Maximization. In *CIKM*, 509–518.

Domingos, P., and Richardson, M. 2001. Mining the Network Value of Customers. In *KDD*, 57–66.

Goyal, A.; Lu, W.; and Lakshmanan, L. V. S. 2011. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *WWW*, 47–48.

Jiang, Q.; Song, G.; Cong, G.; Wang, Y.; Si, W.; and Xie, K. 2011. Simulated Annealing Based Influence Maximization in Social Networks. In *AAAI*, 127–132.

Jung, K.; Heo, W.; and Chen, W. 2012. IRIE: Scalable and Robust Influence Maximization in Social Networks. In *ICDM*, 918–923.

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. Maximizing the Spread of Influence through a Social Network. In *KDD*, 137–146.

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J.; and Glance, N. 2007. Cost-effective Outbreak Detection in Networks. In *KDD*, 420–429.

Nemhauser, G.; Wolsey, L.; and Fisher, M. 1978. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming* 14:265–294.

Richardson, M., and Domingos, P. 2002. Mining Knowledge-Sharing Sites for Viral Marketing. In *KDD*, 61–70.

Sheldon, D.; Dilkina, B.; Elmachtoub, A.; Finseth, R.; Sabharwal, A.; Conrad, J.; Gomes, C.; Shmoys, D.; Allen, W.; Amundsen, O.; and Vaughan, B. 2010. Maximizing the Spread of Cascades Using Network Design. In *UAI*, 517–526.

Wang, Y.; Cong, G.; Song, G.; and Xie, K. 2010. Community-based Greedy Algorithm for Mining Top-K Influential Nodes in Mobile Social Networks. In *KDD*, 1039–1048.