Influence Maximization in Big Networks: An Incremental Algorithm for Streaming Subgraph Influence Spread Estimation

Wei-Xue Lu[†], Peng Zhang[‡], Chuan Zhou[*], Chunyi Liu[*], Li Gao[*]
[*]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[†]Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China
[‡] Center on Quantum Computation & Intelligent Systems, Univ. of Technology Sydney, Australia
wxlu@amss.ac.cn, {zhouchuan@, liuchunyi@nelmail.,gaoli@}iie.ac.cn

Abstract

Influence maximization plays a key role in social network viral marketing. Although the problem has been widely studied, it is still challenging to estimate influence spread in big networks with hundreds of millions of nodes. Existing heuristic algorithms and greedy algorithms incur heavy computation cost in big networks and are incapable of processing dynamic network structures. In this paper, we propose an incremental algorithm for influence spread estimation in big networks. The incremental algorithm breaks down big networks into small subgraphs and continuously estimate influence spread on these subgraphs as data streams. The **challenge** of the incremental algorithm is that subgraphs derived from a big network are not independent and MC simulations on each subgraph (defined as *snapshots*) may conflict with each other. In this paper, we assume that different combinations of MC simulations on subgraphs generate independent samples. In so doing, the incremental algorithm on streaming subgraphs can estimate influence spread with fewer simulations. Experimental results demonstrate the performance of the proposed algorithm.

1 Introduction

Social networks is of vital importance in information diffusion and viral marketing. Influence maximization in social networks is defined as finding a subset of nodes that can trigger the largest number of users propagating the given information. The problem can be formulated as a discrete optimization problem under the independent cascade model (IC) and the linear threshold model (LT) [Kempe *et al.*, 2003]. It has been proved NP-hard, and a greedy algorithm is favored. However, heavy Monte-Carlo simulations are needed to estimate the influence spreads of different seed sets at each iteration. As social networks increase in size continuously, greedy algorithms are inapplicable of processing big networks.

Many efforts have been made to explore scalable algorithms in big netowrks. An intuitive idea is to use heuristic algorithms, i.e. DegreeDiscount [Chen *et al.*, 2009], PMIA [Chen *et al.*, 2010], IRIE [Jung *et al.*, 2011], GroupPageRank [Liu *et al.*, 2014]. However, these algorithms are not robust with respect to network structures. On the other hand, many advanced greedy algorithms, i.e. [Leskovec *et al.*, 2007], [Goyal *et al.*, 2011], [Zhou *et al.*,], have been proposed to speedup seed selection. These methods focus on pruning unnecessary Monte-Carlo simulations in selecting new influential nodes. Unfortunately, these algorithms are still incapable of processing big networks with hundreds of millions of nodes because of unavoidable cost of simulating influence cascades. Moreover, all these greedy methods are designed on static networks and cannot be generalized to dynamic networks which commonly occur in reality.

In this paper, we propose an incremental algorithm for estimating the expected influence spread of seed nodes and evaluate the performance under influence maximization. Specifically, the algorithm breaks down a big network into a large number of small subgraphs, and then processes these subgraphs as data streams. The results on subgraphs are joined to recover the global result in the whole network. The nature of incremental processing also enables it to handle dynamic networks where network structures change over time.

The main challenge of the problem is that small subgraphs derived from a big network are not independent. For example, as in Figure 1, the two subgraphs on the right hand side share nodes 4, 5, 7 and even worse, two subgraphs may share common edges. In this case, MC simulation results on subgraphs (defined as *snapshots*) may not be directly addable or even conflict with each other. In this work, we assume that edges between different subgraphs are disjoint and each subgraph is converted as a class of *Strongly Connected Components* (SCCs). SCCs on subgraphs are joined to form snapshots or SCCs of the whole network, and different combinations of simulations on subgraphs generate independent samples. In experiments, we demonstrate that the solution outperforms heuristics and existing MC simulation based methods.

The merits of the proposed incremental algorithm are summarized as follows:

- A big network is broken down into subgraphs and the "coin flip" technique [Kempe *et al.*, 2003] is applied to generate snapshots on each subgraph. Subgraphs can be processed in parallel.
- Snapshots on subgraphs are converted to be a class of

Strongly Connected Components (SCCs). Snapshots on different subgraphs are joined to restore the global result of the original network.

- The number of snapshot joins increases polynomially with the number of subgraphs, which significantly reduces the complexity of MC simulations.
- Snapshots are incrementally maintained and updated such that replicated simulations can be avoided when changing seed sets.

2 Related Work

Data intensive applications such as data streams motivate incremental learning algorithms. Incremental learning is advantageous when dealing with very large or non-stationary data. Existing incremental learning algorithms can be categorized into two groups: approximate incremental learning [Kivinen *et al.*, 2004] and exact incremental learning [Laskov *et al.*, 2006]. Typical examples include the incremental SVM [Poggio, 2001] and PCA models [Balsubramani *et al.*, 2013]. However, these models were proposed to handle vectorial training data with the Identical and Independent Distributions (IID) assumption. Incrementally learning from subgraph data without the IID assumption has not been addressed before.

In the domain of network influence maximization, most existing models cannot handle big networks with hundreds of millions of nodes due to heavy Monte-Carlo computation. The typical algorithm Cost-Effective Lazy Forward (CELF) [Leskovec *et al.*, 2007] greatly reduce the number of influence spread estimations and is 700 times speed-up against previous greedy algorithms. Unfortunately, these improved greedy algorithms are still inefficient due to too many Monte-Carlo simulations for influence spread estimation. StaticGreedyDU [Cheng *et al.*, 2013] reuses snapshots generated from a given graph and reduces the number of simulations by two orders of magnitude, meanwhile the performance is almost the same as the original greedy algorithm. Recent work [Ohsaka *et al.*, 2014] includes a Pruned BFS method and community-based methods, i.e. [Wang *et al.*, 2010], [Chen *et al.*, 2014]. Unlike these work, our method scales well to big networks by breaking down the big network into small subgraphs for fast spread estimation.

3 Preliminaries

Let $G = (V, E)$ be an undirected graph with a node set V of size $|V|$ and an edge set E of size $|E|$. Without loss of generality, we use the *independent cascade* (IC) model as the propagation model. A node subset $S \subseteq V$ is called a *seed set* if we first activate nodes in S. Given a seed set S, its influence spread $\sigma(S)$ is defined as the expected number of nodes eventually activated. Formally, the influence maximization problem is to find the optimal seed set S^* such that $S^* = \text{argmax}_{S \subseteq V; |S|=k} \sigma(S)$, where $|S|$ stands for the number of elements in S, and k is a predefined integer.

To solve the influence estimation problem, one needs to estimate $\sigma(S)$ for any given S. However, exact computation of $\sigma(\cdot)$ is #P-hard [Chen *et al.*, 2010], so in practice Monte-Carlo simulations are employed to estimate $\sigma(S)$:

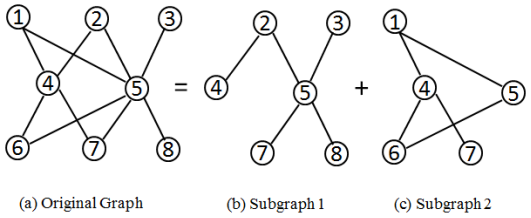


Figure 1: An illustration of the network decomposition.

- Propagation Simulation. The influence spread is obtained by directly simulating the random process propagating from a given seed set S. Let A_i denote the set of nodes newly activated in the i-th iteration and we have $A_0 = S$. In the $i+1$-th iteration, each node $u \in A_i$ has a single chance to activate its each inactive neighbor v with probability p_{uv}. If v is activated, it is added to A_{i+1}. The process stops when $A_{T+1} = \varnothing$ the first time for some T. And $|\widetilde{X}(S)| = |A_0 \cup A_1 \cup \cdots A_T|$ is the influence spread of this single simulation. We run such simulations for many times and obtain the estimated expected influence spread $\sigma(S)$ by averaging over all simulations.
- Snapshot Simulation. Based on "coin flip" [Kempe *et al.*, 2003], we flip coins in advance to produce a *snapshot* $X = (V, E')$, which is a sampled subgraph of G, where an edge uv is either kept with the probability p_{uv} or removed, such that $E' \subseteq E$. Thus, in this specific snapshot X the influence spread for the given seed set S, denoted as $|X(S)|$, equals to the number of nodes reachable from S. We generate a large number of snapshots and obtain the estimated expected influence spread $\sigma(S)$ by averaging over all the generated snapshots.

The above two simulation methods are equivalent, because $|\widetilde{X}(S)|$ and $|X(S)|$ share the same probability distribution [Kempe *et al.*, 2003]. In this work, we use snapshot simulations because we need to frequently estimate the influence spread of different seed sets when choosing new seeds to S^*.

4 Problem Formulation

The basic idea of the paper is to break down a big network into small subgraphs and analyze the reachability of nodes on each subgraph independently. The results from each subgraph are joined to restore the results on the original network.

We use Figure 1 to explain the incremental subgraph algorithm. Figure 1(a) is the original network that each edge is associated with a propagation probability $p_{uv}, u, v \in \{1, 2, \cdots, 8\}$. For simplicity, assume the edges in the graph are split into two disjoint sets and two subgraphs are obtained as shown in Figures 1(b) and (c). The propagation probability p_{uv} on each edge in the subgraphs is inherited from the original network. This way, the original graph is broken down into two small and tractable subgraphs and incremental learning can be applied to the subgraphs.

Consider that the original network $G = (V, E)$ is broken down into N subgraphs $G_1 = (V_1, E_1), G_2 =$

$(V_2, E_2), \cdots, G_N = (V_N, E_N)$ with $V_1 \cup V_2 \cup \cdots \cup V_N = V$, $E_1 \cup E_2 \cup \cdots \cup E_N = E$, and $E_i \cap E_j = \varnothing, 1 \leqslant i, j \leqslant N$. In dynamic networks with increasing nodes and edges, we keep increasing nodes and edges as new streaming subgraphs denoted by $G_{N+1} = (V_{N+1}, E_{N+1}), \cdots$. Then, we use the "coin flip" method on each subgraph G_i for M times and generate M snapshots. Denote these snapshots on G_i as $X_1^{(i)}, X_2^{(i)}, \cdots, X_M^{(i)}$. Then, our problem turns to *given a seed set in a big network, incrementally learning snapshots generated from N subgraphs to estimate the influence spread of the seed set in the original big network.*

5 The Incremental Algorithm

5.1 Expression of Influence

First, each snapshot X_r of the original network can be represented by a m-dimensional binary vector $r = (r_{uv})_{uv \in E}$, where $r_{uv} = 1$ if edge uv is activated in a coin flipping; otherwise, $r_{uv} = 0$. We call r a representation vector of X_r. The closed set of all possible snapshot representation vectors is denoted by R as follows,

$$R = \{r | r = (r_{uv})_{uv \in E} \in \{0, 1\}^{|E|}\} \tag{1}$$

Under the IC model assumption, the probability distribution on R is given by

$$q(r) = \prod_{uv \in E} (p_{uv})^{r_{uv}} (1 - p_{uv})^{1 - r_{uv}},\ r \in R \tag{2}$$

If the original graph is broken down into N subgraphs, the snapshot we get by restricting X_r on subgraph G_i is denoted by $X_r^{(i)} = X_r|_{G_i}$ with the representation vector $r^{(i)} = (r_{uv})_{uv \in E_i}$. Obviously, the marginal distribution of $r^{(i)}$ is $q(r^{(i)}) = \prod_{uv \in E_i} (p_{uv})^{r_{uv}} (1 - p_{uv})^{1 - r_{uv}}$, which similarly describes the distribution of snapshots on subgraph G_i. These distributions satisfy the following equation $q(r) = q(r^{(1)}) q(r^{(2)}) \cdot \cdots \cdot q(r^{(N)})$. Therefore, subgraphs can be processed incrementally by using the network decomposition.

We introduce the symbol "$\oplus$" as the join operator in the equation $X_r = X_r^{(1)} \oplus X_r^{(2)} \oplus \cdots \oplus X_r^{(N)}$, which means that X_r can be broken down into N snapshots $X_r^{(i)}, 1 \leqslant i \leqslant N$, where $X_r^{(i)}$ is the snapshot corresponding to subgraph G_i. Conversely, if snapshots $X_r^{(i)}, 1 \leqslant i \leqslant N$ are increasingly obtained from subgraphs, we can join them to restore the snapshot X_r of the original network. Hence the equation $X_r = X_r^{(1)} \oplus X_r^{(2)} \oplus \cdots \oplus X_r^{(N)}$ interprets the relationship between X_r and $X_r^{(i)}$. Let $N(S; X_r)$ denote the set of nodes reachable from the seed set S in a specific snapshot X_r. The expected influence spread $\sigma(S)$ of the seed set S can be described as follows,

$$\sigma(S) = \sum_{r^{(1)}} q(r^{(1)}) \sum_{r^{(2)}} q(r^{(2)}) \cdots \sum_{r^{(N)}} q(r^{(N)}) \cdot |N(S; X_r)| \tag{3}$$

We can observe from the above equation that $\sigma(S)$ is determined by $|N(S; X_r)| = |N(S; X_r^{(1)} \oplus X_r^{(2)} \oplus \cdots \oplus X_r^{(N)})|$. Thus, we explain how to use the snapshots generated from these N subgraphs to estimate the expected influence spread of a given seed set in the original network.

Algorithm 1: The join of SCC classes.

Input: $\mathcal{C}_1 = \{C_{11}, C_{12}, \cdots, C_{1\alpha_1}\}$ and $\mathcal{C}_2 = \{C_{21}, C_{22}, \cdots, C_{2\alpha_2}\}$
Output: A class of SCCs after joining of $\mathcal{C}_1$ and $\mathcal{C}_2$

```
function join(C_1, C_2)
for j = 1, 2, ..., |C_2| = α_2 do
    s = 1, C ← ∅, C ← ∅;
    for i = 1, 2, ..., |C_1| do
        if C_1i ∩ C_2j ≠ ∅ then
            if C_2j ⊆ C_1i then return C_1;
            else C ← C ∪ C_1i;
        else
            C_s ← C_1i; C ← {C, C_s}; s ← s + 1
    C ← {C, C}; C_1 ← C
return C_1
```

For any snapshot X in the network $G = (V, E)$, we have $u \to v$ if node v is reachable from node u on the snapshot X. We say the two nodes u and v are equivalent ($u \leftrightarrow v$) if and only if both $u \to v$ and $v \to u$ are satisfied. Also, we define that each node u is equivalent to itself, $u \leftrightarrow u$.

Based on the definitions, a set of nodes $C = C(u; X) = \{v | v \leftrightarrow u \text{ on } X, v \in V\}$ is called a *Strongly Connected Component* (SCC) of X represented by the node u. The set C is either a node $\{u\}$ or equivalent nodes. This way, the snapshot X can be described by a class of SCCs $\mathcal{C} = \{C(u; X), u \in E\} = \{C_1, C_2, \cdots\}$, where $C_1, C_2, \cdots$ are disjoint sets, and $C_1 \cup C_2 \cup \cdots = E$. Also, SCCs are used for memory-saving and efficient calculation.

As we aim to find the most influential nodes, single sets in class $\mathcal{C}$ can be removed. Then, $\mathcal{C} = \{C_1, C_2, \cdots, C_\alpha\}$ with $|C_i| \geq 2, \forall i \in \{1, 2, \cdots, \alpha\}$. And the elements of $N(S; X_r)$ can be described as follows,

$$N(S; X_r) = \bigcup_{u \in S} C(u; X_r) \approx \bigcup_{\substack{u \in S \\ |C(u; X_r)| \geqslant 2}} C(u; X_r^{(1)} \oplus \cdots \oplus X_r^{(N)})$$

Based on the above equation, we join SCCs on subgraphs to restore the result on the original network.

5.2 Join of the SCC Classes

Consider that we have transformed all the snapshot $X_j^{(i)}$, $1 \leqslant i \leqslant N, 1 \leqslant j \leqslant M$ generated from N subgraphs into classes of SCCs $\mathcal{C}_j^{(i)} = \{C_{jk}^{(i)},\ 1 \leqslant k \leqslant \alpha_j^{(i)},\ |C_{jk}^{(i)}| \geqslant 2\}$, where $\mathcal{C}_j^{(i)}$ is a SCC class of $X_j^{(i)}$. We have discussed that if the snapshots have been incrementally obtained on each subgraph, we can join them to obtain snapshots of the original network. Now, the problem turns to using the SCC classes of these N subgraphs to restore SCC classes of the original network. What we need to do next is to join the SCC classes of these subgraphs. We take $N = 4$ for example. Suppose we incrementally have $\mathcal{C}_{j_1}^{(1)}, \mathcal{C}_{j_2}^{(2)}, \mathcal{C}_{j_3}^{(3)}, \mathcal{C}_{j_4}^{(4)}$, where each $\mathcal{C}_{j_i}^{(i)}$ is a SCC class of the subgraph G_i. In order to restore a SCC class of the original network, we can first join $\mathcal{C}_{j_1}^{(1)}, \mathcal{C}_{j_2}^{(2)}$ to obtain $\mathcal{C}_{j_{1,2}}^{(1,2)}$ which is a SCC class of the subgraph $G_1 \oplus G_2 \triangleq (V_1 \cup V_2, E_1 \cup E_2)$, then $\mathcal{C}_{j_3}^{(3)}$ arrives and we increasingly join it with $\mathcal{C}_{j_{1,2}}^{(1,2)}$ to ob-

tain $\mathcal{C}_{j_{1,2,3}}^{(1,2,3)}$. At last, $\mathcal{C}_{j_4}^{(4)}$ arrives and we join it with $\mathcal{C}_{j_{1,2,3}}^{(1,2,3)}$ to obtain $\mathcal{C}_{j_{1,2,3,4}}^{(1,2,3,4)}$, which is a SCC class of the original network. From this example, we can observe that the join of $\mathcal{C}_{j_1}^{(1)}, \mathcal{C}_{j_2}^{(2)}, \cdots, \mathcal{C}_{j_N}^{(N)}, 1 \leqslant j_1, \cdots, j_N \leqslant M$ can be reduced to the join of any two SCC classes.

Therefore, we define a function $\text{join}(\mathcal{C}_1, \mathcal{C}_2)$ in Algorithm 1 to return the class after joining $\mathcal{C}_1 = \{C_{11}, C_{12}, \cdots, C_{1\alpha_1}\}$ and $\mathcal{C}_2 = \{C_{21}, C_{22}, \cdots, C_{2\alpha_2}\}$.

5.3 Influence Estimation

In this section, we first briefly introduce the traditional network influence estimation methods theoretically guaranteed by Theorem 1. When extending the traditional methods into streaming subgraphs, we can join the snapshots of subgraphs to restore samples of the original network. However, the join of subgraph snapshots exponentially increases with the number of subgraphs. So, we propose to select only a few number of snapshot joins. Proposition 3 explains that when the selected joins are independent and identically distributed (i.i.d.), the estimation converges fast. However, it is hard to select i.i.d. snapshot joins across subgraphs in reality. Therefore, we propose Definitions 1 and 2 to choose snapshot joins that approximately follow the i.i.d. condition. Proposition 3 shows that by using the approximation, we can use a polynomial number of snapshot joins to restore the global result.

For simplicity, $|N(S; X_{j_1}^{(1)} \oplus X_{j_2}^{(2)} \oplus \cdots \oplus X_{j_N}^{(N)})|$ is denoted by $g_{j_1,j_2,\cdots,j_N}(S)$, which means that the snapshot of the original network can be obtained by incrementally joining the j_i-th snapshot from the i-th subgraph, and $g_{j_1,j_2,\cdots,j_N}(S)$ returns the influence spread of a given seed set S. The representation vector of $X_{j_1}^{(1)} \oplus X_{j_2}^{(2)} \oplus \cdots \oplus X_{j_N}^{(N)}$ follows the distribution $q(r)$. Hence for any $1 \leqslant j_1, \cdots, j_N \leqslant M$, $g_{j_1,j_2,\cdots,j_N}(S)$ is a random variable such that $\mathbf{E} g_{j_1,j_2,\cdots,j_N}(S) = \sigma(S)$, where $\mathbf{E}$ is the expectation operator. Generally, M simulations are independently employed on the original network, generating M snapshots denoted by $X_i = X_i^{(1)} \oplus \cdots \oplus X_i^{(N)}$, and $\sigma(S)$ is empirically estimated by the following equation $\widehat{\sigma}(S) = \frac{1}{M}\sum_{i=1}^{M} g_{i,i,\cdots,i}(S)$ where $g_{i,i,\cdots,i}(S)$, $1 \leq i \leq M$ are bounded and i.i.d.. So the estimation is guaranteed by the following Theorem [Hoeffding, 1963],

Theorem 1 (Hoeffding's inequality). *Let $Y_1, \cdots, Y_M$ be independent variables span over the interval $[a, b]$, and $S_M \triangleq Y_1 + \cdots + Y_M$. Then, for all $\varepsilon > 0$,*

$$\mathbf{P}\left(\frac{1}{M}|S_M - \mathbf{E}S_M| \geqslant \varepsilon\right) \leqslant 2\exp\left(-\frac{2M\varepsilon^2}{(b-a)^2}\right) \quad (4)$$

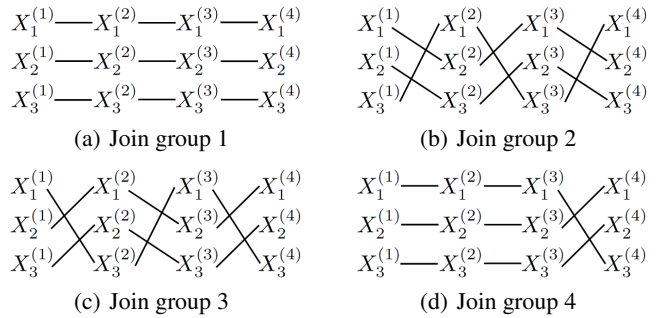
In the traditional problem setting, components of the subscript of $g_{i,i,\cdots,i}(S)$ are the same, because simulations are employed on a single network G and subscript $(i, i, \cdots, i)$ denotes the i-th snapshot simulation on G. Furthermore, snapshots on subgraphs are not obtained by independent simulations but by restricting snapshots of the original network on subgraphs. In contrast, our incremental algorithm considers that snapshots of subgraphs are generated independently and then joined to the previous results. Specifically, we first generate M snapshots $X_{j_1}^{(1)}, 1 \leqslant j_1 \leqslant M$ on subgraph G_1, then the subgraph G_2 arrives where we generate another M snapshots $X_{j_2}^{(2)}, 1 \leqslant j_2 \leqslant M$. We join them to the snapshots of G_1 and obtain snapshots $X_{j_1}^{(1)} \oplus X_{j_2}^{(2)}$. The process continues until the N-th subgraph is joined. This way, we generate M^N random variables $g_{j_1,j_2,\cdots,j_N}(S)$, $1 \leqslant j_1, j_2, \cdots, j_N \leqslant M$ such that $\mathbf{E} g_{j_1,j_2,\cdots,j_N}(S) = \sigma(S)$. Unfortunately, it is not reasonable to take $S_\Sigma \triangleq 1/M^N \sum_{j_1=1}^{M} \cdots \sum_{j_N=1}^{M} g_{j_1,j_2,\cdots,j_N}(S)$ as an estimation of $\sigma(S)$. On the one hand, the number of snapshot joins M^N exponentially increases with respect to the number of subgraphs. On the other hand, many snapshot joins are redundant when estimating the expected influence spread. To better explain these two points, we first rewrite S_Σ as the average summation of M^{N-1} groups such that each group is the average summation of M independent and identically distributed random variables, i.e., $S_\Sigma = 1/M^{N-1} \sum_{\beta=1}^{M^{N-1}} \left[1/M \sum_{\alpha=1}^{M} g_{j_1^{(\alpha,\beta)},\cdots,j_N^{(\alpha,\beta)}}(S)\right] = 1/M^{N-1} \sum_{\beta=1}^{M^{N-1}} T_\beta$, where $g_{j_1^{(\alpha,\beta)},\cdots,j_N^{(\alpha,\beta)}}(S), 1 \leqslant \alpha \leqslant M$ in T_β are i.i.d.. Figure 2(a) shows the join of three snapshots of the original network and the first join snapshot is obtained by incrementally join $X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}$. This join snapshot corresponds to the random variable $g_{1,1,1,1}(S)$. Similarly, the rest two corresponding random variables are $g_{2,2,2,2}(S)$ and $g_{3,3,3,3}(S)$. It is clear that these three are i.i.d. and the average summation is $T_1 = 1/3\left[g_{1,1,1,1}(S) + g_{2,2,2,2}(S) + g_{3,3,3,3}(S)\right]$. Hence, Figure 2(a) corresponds to T_1. Similarly, Figure 2(b), 2(c), 2(d) correspond to T_2, T_3, T_4 respectively. Note that not all T_j are shown in the figure. Each T_j can be treated as some S_M in Theorem 1 and Eq. (4) is satisfied, i.e., $\mathbf{P}\left(|T_j - \sigma(S)| \geq \varepsilon\right) \leqslant 2\exp\left(-\frac{2M\varepsilon^2}{(b-a)^2}\right)$. Therefore, each T_j has the same convergence rate as the traditional method. Next, our goal is to choose a portion of T_j to estimate the influence spread in order to obtain higher convergence rate. If the same convergence rate is required, our method needs fewer simulations on each subgraph than the traditional method. Moreover, our method can deal with dynamic networks.

For this goal, we first introduce the following proposition.

Proposition 2. *If $T_1, \cdots, T_k$ are independent and identically distributed random variables, then $1/k \sum_{i=1}^{k} T_i$ has a higher convergence rate than the righthand side of Eq. (4).*

Proof. By applying the independence and Hoeffding's lemma [Hoeffding, 1963], we have, for any $t > 0$

$$\begin{aligned}
&\mathbf{P}\left(\frac{1}{k}\sum_{i=1}^{k} T_i - \sigma(S) \geqslant \varepsilon\right) = \mathbf{P}\left(t\frac{1}{k}\sum_{i=1}^{k} T_i - t\sigma(S) \geqslant t\varepsilon\right) \\
\leqslant &\mathbf{E}\exp\left(t\frac{1}{k}\sum_{i=1}^{k} T_i\right) e^{-t\sigma(S)} e^{-t\varepsilon} = \prod_{i=1}^{k} \mathbf{E} e^{\frac{t}{k}T_i} e^{-t\sigma(S)} e^{-t\varepsilon} \\
\leqslant &\prod_{i=1}^{k} \exp\left(\frac{t}{k}T_i\right) \exp\left(\frac{1}{8M}\frac{t^2}{k^2}(b-a)^2\right) e^{-t\sigma(S)} e^{-t\varepsilon} \\
= &\exp\left(\frac{1}{8M}\frac{t^2}{k}(b-a)^2 - t\varepsilon\right) \leqslant \exp\left(\frac{-2 \cdot kM\varepsilon^2}{(b-a)^2}\right)
\end{aligned}$$



(a) Join group 1 (b) Join group 2

(c) Join group 3 (d) Join group 4

Figure 2: Joins of subgraph snapshots. $X_j^{(i)}$ represents the j^{th} snapshot simulation result on the i^{th} subgraph.

The first inequality is based on the Chebyshev's inequality. The second equality is guaranteed by the independence assumption of $T_1, \cdots, T_k$. The second inequality is due to the Hoeffding's lemma. The last inequality is the upper bound of the previous term after maximization with respect to t. The last item of the above equation shows that a higher convergence rate is obtained. □

From the above proposition, the strategy is to choose those T_j that tend to be independent. Besides, the influence spread of a particular seed set is not only dependent on different snapshots $X_j^{(i)}$ of subgraphs but also on different joins of them, which determines the network topology of the snapshot of the original graph. Hence, we define $(j_1, \cdots, j_N)$ as a join path with $N-1$ stages $(j_1, j_2), (j_2, j_3), \cdots, (j_{N-1}, j_N)$.

Definition 1. *We call two join paths* $(i_1, i_2, \cdots, i_N)$ *and* $(j_1, j_2, \cdots, j_N)$ *are* orthogonal *if there is no overlapping stage, i.e.,* $(i_\alpha, i_{\alpha+1}) \neq (j_\alpha, j_{\alpha+1})$ *for all* $1 \leqslant \alpha \leqslant N-1$.

Definition 2. *Let* $T_{\beta_1} = 1/M \sum_{\alpha=1}^{M} g_{j_1^{(\alpha,\beta_1)}, \cdots, j_N^{(\alpha,\beta_1)}}$ *and* $T_{\beta_2} = 1/M \sum_{\alpha'=1}^{M} g_{j_1^{(\alpha',\beta_2)}, \cdots, j_N^{(\alpha',\beta_2)}}$.$T_{\beta_1}$ *and* T_{β_2} *are called* join-independent *if for any given* α *and* α' *such that* $1 \leqslant \alpha', \alpha \leqslant M$, $(j_1^{(\alpha,\beta_1)}, \cdots, j_N^{(\alpha,\beta_1)})$ *and* $(j_1^{(\alpha',\beta_2)}, \cdots, j_N^{(\alpha',\beta_2)})$ *are orthogonal.*

For example, in Figure 2, T_1, T_2, T_3 are pairwise join-independent and there are paths in T_1 and T_4 with two overlapping stages. Now, we introduce the set $\mathcal{T} = \{T_\beta | T_{\beta_1}, T_{\beta_2}$ are join-independent, $\forall T_{\beta_1}, T_{\beta_2} \in \mathcal{T}\}$. We use $T_\beta \in \mathcal{T}$ to estimate the influence spread. The following proposition shows how many T_β are selected.

Proposition 3. *Let* $\mathcal{T}$ *be the set of* $\mathcal{T}_\beta$ *that are join-dependent pairwise, i.e.,* $\mathcal{T} = \{T_\beta | T_{\beta_1}, T_{\beta_2}$ *are join-independent,*$\forall T_{\beta_1}, T_{\beta_2} \in \mathcal{T}\}$*, then we have* $|\mathcal{T}| = M$.

Proof. All the paths in $T_\beta \in \mathcal{T}$ form a set of paths $\mathcal{P}$. Each candidate $\mathcal{T}$ corresponds to a unique $\mathcal{P}$. All candidates $\mathcal{T}$ (or $\mathcal{P}$) have the same cardinality. It can be verified that the set $\mathcal{P}_0 = \{(i, j, i, j, \cdots), 1 \leqslant i, j \leqslant M\}$ is a portion of candidates $\mathcal{P}$. So, we have $|\mathcal{P}| = |\mathcal{P}_0| = M^2$, which means $|\mathcal{T}| = M$ for $T_\beta \in \mathcal{T}$ with M paths. □

Algorithm 2: The Subgraph Incremental Algorithm.

Input: $G = (V, E), G_1 = (V_1, E_1), \cdots, G_N = (V_N, E_N)$, k, R, propagation probability p
Output: the optimal seed set S that maximizes the influence on the original network

```
for i = 1, 2, ..., N do
    for j = 1, 2, ..., M do
        sample each edge with probability p on the i-th subgraph to obtain a snapshot X_j^(i);
        use the DFS graph search to get a class of SCCs C_j^(i) ;
for i = 1, 2, ..., M do
    for j = 1, 2, ..., M do
        calculate C_ij using the function join(·, ·)
S ← ∅;
while |S| < k do
    for v ∈ V\S do
        σ(v|S) ← 0;
        for i = 1, 2, ..., M do
            for j = 1, 2, ..., M do
                if v ↔ S then return σ(v|S);
                else σ(v|S) ← σ(v|S) + |C_ij(v)|;
    t ← argmax_{v∈V\S} (1/M^2) σ(v|S);
    S ← S ∪ {t}
return S
```

Therefore, we regard $\widehat{\sigma}(S) = 1/M \sum_{T_\beta \in \mathcal{T}} T_\beta$ as the estimation of the influence spread given the seed set S on the original graph. Though there may be many such $\mathcal{T}$, from the proof of Proposition 3 it is easily can be seen that there exists some $\mathcal{T}_0$ such that the set all the paths in $T_\beta \in \mathcal{T}_0$ equals $\mathcal{P}_0$. Hence one specific expression of $\widehat{\sigma}(S)$ can be given by $\widehat{\sigma}(S) = 1/M \sum_{i=1}^{M} \sum_{j=1}^{M} g_{i,j,i,j,\cdots}(S)$, where a polynomial number M^2 of snapshot joins are used.

5.4 Influence Maximization

In this section, we evaluate our estimation of the influence spread in the process of influence maximization. Greedy algorithm [Kempe *et al.*, 2003] is adopted to select seed nodes, starting with an empty seed set S, and then add the node u with the maximum marginal influence in every iteration, i.e., $u = \text{argmax}_{v \in V \setminus S} \sigma(S \cup v) - \sigma(S)$, into S until the threshold k ($|S| = k$) is met. Because the influence spread function is non-negative, monotone and submodular, the accuracy of the optimum solution is guaranteed by $f(S) \geqslant (1 - 1/e) f(S^*)$, where S^* is the optimal solution [Nemhauser *et al.*, 1978]. We summarize the above in Algorithm 2.

Note that $\widehat{\sigma}(S)$ is used in each iteration when the marginal influence is calculated $\widehat{\sigma}(S \cup v) - \widehat{\sigma}(S) = 1/M^2 \sum_{i=1}^{M} \sum_{j=1}^{M} [g_{i,j,i,j,\cdots}(S \cup v) - g_{i,j,i,j,\cdots}(S)] = 1/M^2 \sum_{i=1}^{M} \sum_{j=1}^{M} \{|N(S \cup v; X_i^{(1)} \oplus X_j^{(2)} \oplus \cdots)| - |N(S; X_i^{(1)} \oplus X_j^{(2)} \oplus \cdots)|\} = 1/M^2 \sum_{i=1}^{M} \sum_{j=1}^{M} [1 - I(S \leftrightarrow v)] |C_{i,j}(v)|$, where $I(S \leftrightarrow v) = 1$ if $\exists\ s \in S$ such that $s \leftrightarrow v$, $C_{i,j}(v)$ denotes the SCC including v on the snapshot

Datasets	number of nodes	number of edges
ego-Facebook	4,039	176,468
ca-HepPh	12,008	118,521
ca-CondMat	23,133	186,936
email-Enron	36,692	367,662

Table 1: The four real-world datasets.

$X_i^{(1)} \oplus X_j^{(2)} \oplus \cdots$ whose class of SCCs is denoted by $\mathcal{C}_{ij}$.

6 Experiments

We conduct experiments on four real-world data sets. All algorithms are implemented in C++ and run on a Windows 7 system with 2.3GHz CPU and 4GB memory.

6.1 Data Sets

We use four network datasets from *snap.stanford.edu/data/* for testing and comparisons. The number of nodes and edges in each network is summarized in Table 1.

- **ego-Facebook:** This data set consists of 'circles' (or 'friends lists') from Facebook. It was collected from survey participants using the Facebook app.
- **ca-HepPh** and **ca-CondMat:** These two data sets are obtained from the e-print arXiv. If authors i and j publish a paper together, the network contains an edge connecting nodes i to j.
- **email-Enron:** Nodes of the network are email addresses and if an address i sent at least one email to address j, the graph contains an undirected edge from i to j.

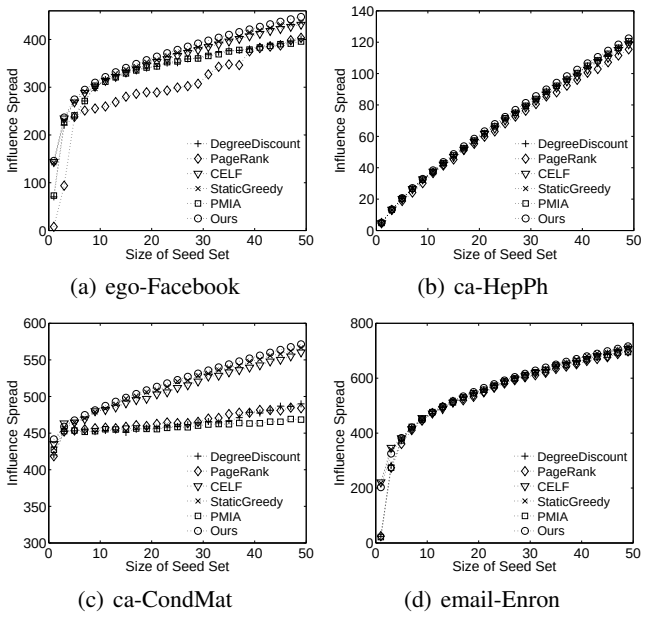
6.2 Benchmark Methods

We compare the incremental algorithm with five algorithms. The propagation probability of each edge is set to $p = p_{uv} = 0.01, uv \in E$.

- **CELF:** We set $M = 10,000$ in the Monte-Carlo simulations to estimate the spread of a seed set.
- **StaticGreedy:** StaticGreedy reuses the generated snapshots to reduce the number of simulations. The number of snapshots is set to $M = 10,000$.
- **DegreeDiscount:** This algorithm is developed for the IC model with the uniform propagation probability.
- **PMIA:** It uses maximization paths for influence spread estimation. The value of the parameter θ is set to $1/320$.
- **PageRank:** We implement the power method with a damping factor of 0.85 and set the k highest-ranked nodes as seeds. The algorithm stops by the threshold value of 10^{-6} with the L_1-norm.

6.3 Experimental Results

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



(a) ego-Facebook (b) ca-HepPh

(c) ca-CondMat (d) email-Enron

Figure 3: Comparisons between benchmark algorithms and the incremental algorithm on the four data sets.

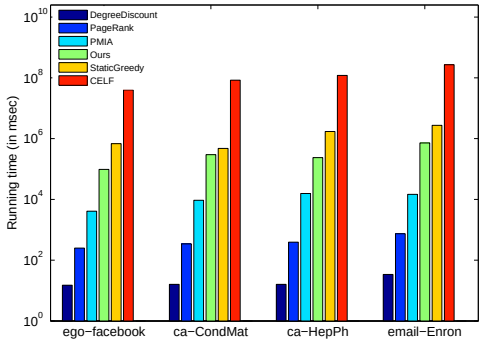


Figure 4: Comparisons *w.r.t.* the running time on the four data sets

Influence spread. Figure 3 shows the influence spreads of different algorithms on the real-world networks with respect to the seed set size k. k increases from 1 to 50. From the figure, we can observe that our method performs the best in all settings. Together with CELF and StaticGreedy, our method is also independent of the network structures. Moreover, only $M = 100$ snapshots on each subgraph are generated to estimate the influence spreads in the process of selecting the most influential nodes. The results show much fewer MC simulations than the other greedy methods.

Running time. Figure 4 shows the running time under the optimal seed set of size 50 on the four datasets. Undoubtedly, heuristic methods, including DigreeDiscount, PMIA and PageRank, are the fastest. CELF is the slowest because too many MC simulations are employed during the estimation of influence spreads. We can observe that the speed of our algorithm is as fast as the StaticGreedy algorithm which

is a popularly used scalable algorithm.

From these result, we can conclude that our algorithm is robust to network structures, performs the best compared with existing algorithms, and scale well to large networks.

7 Conclusions

In this paper, we presented an incremental algorithm for solving the big network influence maximization problem. Our idea is to break down the original graph into small subgraphs which can be sequentially processed as subgraph streams. Experimental results show that the algorithm is robust to network structures, performs the best compared with existing algorithms, and scale well to large networks.

Acknowledgments

This work was supported by the NSFC (No. 61370025), and the Strategic Leading Science and Technology Projects of CAS (No.XDA06030200), 973 project (No. 2013CB329605) and Australia ARC Discovery Project (DP140102206).

References

[Balsubramani *et al.*, 2013] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. The fast convergence of incremental pca. In *Advances in Neural Information Processing Systems*, pages 3174–3182, 2013.

[Chen *et al.*, 2009] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[Chen *et al.*, 2010] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.

[Chen *et al.*, 2014] Yi-Cheng Chen, Wen-Yuan Zhu, Wen-Chih Peng, Wang-Chien Lee, and Suh-Yin Lee. Cim: Community-based influence maximization in social networks. *ACM Trans. Intell. Syst. Technol.*, 5(2):25:1–25:31, April 2014.

[Cheng *et al.*, 2013] Suqi Cheng, Huawei Shen, Junming Huang, Guoqing Zhang, and Xueqi Cheng. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 509–518. ACM, 2013.

[Goyal *et al.*, 2011] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 47–48. ACM, 2011.

[Hoeffding, 1963] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[Jung *et al.*, 2011] Kyomin Jung, Wooram Heo, and Wei Chen. Irie: Scalable and robust influence maximization in social networks. *arXiv preprint arXiv:1111.4795*, 2011.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[Kivinen *et al.*, 2004] Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, 2004.

[Laskov *et al.*, 2006] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *The Journal of Machine Learning Research*, 7:1909–1936, 2006.

[Leskovec *et al.*, 2007] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

[Liu *et al.*, 2014] Qi Liu, Biao Xiang, Enhong Chen, Hui Xiong, Fangshuang Tang, and Jeffrey Xu Yu. Influence maximization over large-scale social networks: A bounded linear approach. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 171–180. ACM, 2014.

[Nemhauser *et al.*, 1978] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsłi. *Mathematical Programming*, 14(1):265–294, 1978.

[Ohsaka *et al.*, 2014] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. 2014.

[Poggio, 2001] Gert CauwenberghsTomaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 409. MIT Press, 2001.

[Wang *et al.*, 2010] Yu Wang, Gao Cong, Guojie Song, and Kunqing Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1039–1048. ACM, 2010.

[Zhou *et al.*,] Chuan Zhou, Peng Zhang, Wenyu Zang, and Li Guo. On the upper bounds of spread for greedy algorithms in social network influence maximization. *IEEE Transactions on Knowledge and Data Engineering, no. 1, pp. 1, PrePrints PrePrints, doi:10.1109/TKDE.2015.2419659*.