2-hop+ Sampling: Efficient and Effective Influence Estimation

Yuqing Zhu, Jing Tang, *Member, IEEE,* Xueyan Tang, *Member, IEEE,* Sibo Wang, and Andrew Lim

Abstract—With rapidly growing sizes of online social networks, computational challenges arise in analyzing the diffusion process over networks. Sampling methods are commonly used to study the cascade effect and estimate users' influence. In this paper, we propose a brand-new sampling method, called 2-hop+ sampling for quickly and accurately estimating the cascade size generated by a set of seed users under the independent cascade model. Our method generates only samples with at least one 2-hop live path from the source to reduce the number of samples. We further enhance the sampling efficiency of our method by a SkipEdge technique. Moreover, we improve the generalized stopping rule algorithm to obtain an (ε, δ)-estimate of the mean of random variables with fewer samples needed. Extensive experiments with real-world datasets show that our techniques can significantly improve the estimation efficiency compared to the state-of-the-art methods.

Index Terms—Online social networks, influence estimation, sampling

1 Introduction

The popularity of online social networks (OSNs) has given birth to the studies on information propagation over OSNs in the past decade. With the word of mouth effects, a small number of users can trigger a large cascade of information propagation through the network. There are many problems about information propagation such as estimating users' influence [13, 25, 29], influence maximization [4, 33, 34, 39, 40], analyzing network centrality [5], and identifying critical nodes in the network [12]. To tackle these problems, a central task is to analyze the diffusion process and predict the cascade size in the network, which is known as influence estimation in marketing campaigns. However, the growing sizes of OSNs nowadays have made this task exceedingly computationally expensive.

Existing work [4, 6, 10, 20, 28, 29, 30, 34, 40] mostly utilizes sampling methods to estimate the cascade size. Early studies use Monte Carlo simulations to generate samples [6, 10, 20, 30]. Recent work [4, 27, 28, 34, 40] adopts the advanced *reverse influence sampling (RIS)* [4] technique. In general, to ensure the accuracy of influence estimation, the computational efficiency of different sampling methods is dependent on the number of samples to generate. Thus, a natural direction to improve efficiency is to reduce the number of samples needed. In this paper, we aim to speed up the RIS approach for influence estimation with accuracy guarantees by less samples.

- *Yuqing Zhu is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: yuqing002@ntu.edu.sg*
- *Jing Tang is with the Data Science and Analytics Thrust, The Hong Kong University of Science and Technology, China. E-mail: tang0311@ntu.edu.sg*
- *Xueyan Tang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: asxytang@ntu.edu.sg*
- *Sibo Wang is with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong. E-mail: swang@se.cuhk.edu.hk*
- *Andrew Lim is with the School of Computer and Artificial Intelligence, Southwest Jiaotong University, China. E-mail: alim@redjasper.com*

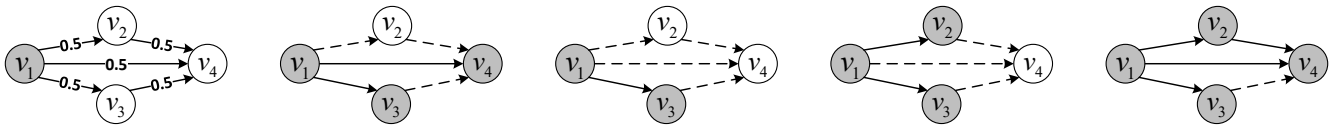
Corresponding author: Jing Tang.

It is known that each additional hop of propagation usually contributes progressively less towards the cascade size [1, 14, 33, 36]. Moreover, the cascade size within 2 hops of propagation can be computed via independent paths [33, 36]. Motivated by these findings, we design a method, called 2-hop+, to generate reverse samples that can spread beyond the source with at least one 2-hop live path. By utilizing such 2-hop samples, our estimations can have higher concentrations with narrower ranges so that less samples need to be generated for achieving a predefined estimation accuracy. Moreover, to speed up the generation of a 2-hop sample, we propose a SkipEdge algorithm that boosts the sampling process by avoiding flipping a coin on each edge based on the dependencies in the sampling process.

In addition, we improve the stopping rule algorithm that can accurately estimate bounded random variables with a theoretically lenient stopping condition in a more elegant expression. In particular, we devise more general concentration bounds by relaxing the restriction that the number of samples must be an integer as required by previous work [11, 29, 40]. Based on the new concentration bounds, we theoretically show that our algorithm requires less samples to return an approximate estimation.

Our contributions can be summarized as follows.

1) We propose a sampling method called 2-hop+ to generate only the samples spreading influence beyond the source with at least one 2-hop live path. The samples generated by our 2-hop+ method can improve influence estimation efficiency with better concentration bounds. We further enhance the sampling efficiency of our method with a SkipEdge technique.
2) We improve the stopping rule algorithm to obtain an (ε, δ)-estimate of bounded random variables with a theoretically tighter threshold by developing generalized concentration bounds.
3) We compare our approaches with the state-of-the-art methods, and demonstrate that our 2-hop+ sampling method is vastly superior in both asymptotic and practical performance.



(a) An Example Graph (b) Possible Cascade 1 (c) Possible Cascade 2 (d) Possible Cascade 3 (e) Possible Cascade 4

Fig. 1: A social graph with 4 nodes and 5 edges and four possible cascades starting from $\{v_1\}$. Dashed arrows indicate blocked edges.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries on influence estimation. We propose the 2-hop+ sampling method in Section 3. We elaborate the improvement on the stopping rule algorithm in Section 4. Section 5 presents the experiment details and Section 6 reviews the related work. Finally, Section 7 concludes the paper.

2 PRELIMINARIES

We model an online social network as a directed graph $G = (V, E)$ comprising a set V of nodes and a set E of directed edges. For each directed edge $(u, v) \in E$, v is known as u's neighbor and u is known as v's inverse neighbor. For each node $u \in V$, let I_u denote the set of node u's inverse neighbors, i.e., $I_u = \{w\colon w \in V, (w, u) \in E\}$. Each edge (u, v) is associated with a propagation probability $p_{u,v}$, representing the probability that u influences v. There are many diffusion models characterizing the influence propagation in the network, among which the independent cascade (IC) model and the linear threshold (LT) model [20] are two most commonly used ones. Under the IC model, each activated node u has a single chance of probability $p_{u,v}$ to activate each inactive neighbor v. Under the LT model, the propagation probabilities are normalized to satisfy $\sum_{u \in I_v} p_{u,v} \leq 1$. The decision to activate a node v is based on a threshold λ_v chosen uniformly and randomly from $[0, 1]$. A node v becomes active if $\sum_{u \in A_v} p_{u,v} \geq \lambda_v$, where $A_v \subseteq I_v$ denotes the set of v's active inverse neighbors. In both models, the diffusion process starts from a given set of activated nodes S, called the seed set, and the remaining nodes are initially inactive. In the diffusion process, once a node becomes activated, it remains activated for all subsequent steps. The process ends when no more node can be activated. Given a diffusion model, the influence spread of a seed set S, denoted as $\sigma(S)$, is the expected number of nodes activated by S when the diffusion process terminates. Computing the influence spread $\sigma(S)$ of a seed set S is #P-hard under both the IC and LT models [7, 8].

Example 1. *Fig. 1(a) shows an example graph with 4 nodes and 5 edges where each edge is associated with a propagation probability of 0.5. Then, under the IC model, there are a total number of $2^5 = 32$ possible cascades, each occurring with a probability of $1/32$. Figs. 1(b)–1(e) show four possible cascades starting from a seed set $\{v_1\}$, where dashed (resp. solid) arrows indicate blocked (resp. live) edges. If the cascade in Fig. 1(b) happens, nodes $\{v_1, v_3, v_4\}$ are activated with an influence of 3. Similarly, the influences in other three cascades are 2, 3 and 4, respectively. Taking the expectation over all 32 possible cascades, we can derive that $\{v_1\}$'s expected influence spread is $\sigma(\{v_1\}) = 2.7$.*

The influence estimation problem is to obtain an estimation $\tilde{\sigma}(S)$ of $\sigma(S)$. The reverse influence sampling (RIS) method proposed by Borgs et al. [4] is the state-of-the-art approach that estimates $\sigma(S)$ by means of sampling. Each sample produced by RIS is known as a random reverse reachable (RR) set which basically contains a set of nodes that can influence a randomly chosen node v. We focus on the IC model in this paper. Under the IC model, a random RR set R can be generated as follows [4].

1) Select a source node v from V uniformly at random.
2) Perform a stochastic breadth first search (BFS) starting from v following the reverse direction of the edges in the graph. For each node u encountered in the BFS, a coin is flipped for each incoming edge (w, u) to determine whether the edge is live. That is, with probability $p_{w,u}$, the edge is set to live and the BFS traverses to w from u if w has not been visited, and with probability $1 - p_{w,u}$, the edge is set to blocked and is ignored by the BFS. All the traversed nodes are inserted into R.

Intuitively, a node has a higher probability to be included in a random RR set if it has greater influence on other nodes. The influence spread can be estimated by a set of RR sets.

Lemma 1 ([4])**.** *Given a seed set $S \subseteq V$, for a random RR set R, we have*

$$\sigma(S) = |V| \cdot \Pr[R \cap S \neq \emptyset],$$

where $|V|$ denotes the total number of nodes in the graph.

Based on Lemma 1, to estimate $\sigma(S)$, we can generate a sequence $\mathcal{R}$ of random RR sets, count the number of RR sets overlapping S, denoted as $\Lambda(\mathcal{R}, S)$, and get an empirical estimate of $\Pr[R \cap S \neq \emptyset]$ by $\frac{\Lambda(\mathcal{R},S)}{|\mathcal{R}|}$. Then, the estimation $\tilde{\sigma}(S)$ is computed as $|V| \cdot \frac{\Lambda(\mathcal{R},S)}{|\mathcal{R}|}$.

3 2-HOP+ SAMPLING

The main motivation for our new 2-hop+ sampling method is that the influence spread within 1 hop of propagation can be computed exactly and efficiently for any seed set [33, 36]. Our key idea is to generate only the random RR sets necessary for estimating the influence spread beyond 1 hop of propagation and then combine the estimation result with the 1-hop influence computed to produce an estimate of the overall influence spread. Since the 1-hop influence can constitute a significant portion of the overall influence spread [33, 36], the random variables in our sampling can have a narrower range with higher concentrations improving the estimation accuracy of the total influence spread. In addition, we propose a new technique, called SkipEdge, to efficiently sample the incoming edges according to their probability distribution.

3.1 2-hop Sampling Algorithm

3.1.1 2-hop RR Sets

Let S be a seed set. Recall that the key to computing S's influence spread is to estimate the probability for a random RR set R to overlap S, i.e., $\Pr[R \cap S \neq \emptyset]$. We decompose an RR set R

into three parts: $\mathrm{src}(R)$, $\mathrm{h}_1(R)$, and $\mathrm{h}_{2^*}(R)$, where $\mathrm{src}(R)$ is the source node of R, $\mathrm{h}_1(R)$ is the set of nodes that activate v directly in the sample, i.e., $\mathrm{h}_1(R) = \{u \mid \text{edge } (u, \mathrm{src}(R)) \text{ is live}\}$, and $\mathrm{h}_{2^*}(R)$ includes all the remaining nodes of R. Then, we have the following three disjoint cases when $R \cap S \neq \emptyset$:

$$\mathcal{E}_a\colon\ \mathrm{src}(R) \in S,$$
$$\mathcal{E}_b\colon\ \mathrm{src}(R) \notin S \text{ and } \mathrm{h}_1(R) \cap S \neq \emptyset,$$
$$\mathcal{E}_c\colon\ \mathrm{src}(R) \notin S, \mathrm{h}_1(R) \cap S = \emptyset \text{ and } \mathrm{h}_{2^*}(R) \cap S \neq \emptyset.$$

By Lemma 1, we have

$$\begin{aligned}\sigma(S) &= |V| \cdot \Pr[R \cap S \neq \emptyset] \\ &= |V| \cdot (\Pr[\mathcal{E}_a] + \Pr[\mathcal{E}_b] + \Pr[\mathcal{E}_c]).\end{aligned} \tag{1}$$

For $\mathcal{E}_a$, since the source node is selected uniformly at random, we have

$$\Pr[\mathcal{E}_a] = \Pr[\mathrm{src}(R) \in S] = \frac{|S|}{|V|}. \tag{2}$$

For $\mathcal{E}_b$, $\mathrm{h}_1(R) \cap S \neq \emptyset$ holds only if the source $\mathrm{src}(R)$ is activated by a seed in S directly. Since each inverse neighbor of a node v activates v independently, the probability for all v's inverse neighbors in S to fail to activate v directly is $\prod_{u \in I_v \cap S}(1 - p_{u,v})$. Therefore, given a source node v, the probability for v to be activated by a seed directly is

$$p_{S,v} = 1 - \prod\nolimits_{u \in I_v \cap S}(1 - p_{u,v}).$$

Let N_u denote the set of node u's neighbors, i.e., $N_u = \{w\colon w \in V, (u, w) \in E\}$. Let $N_S = \bigcup_{u \in S} N_u$ be the set of neighbors of the seed set. Then, the probability for $\mathcal{E}_b$ to happen is given by

$$\begin{aligned}\Pr[\mathcal{E}_b] &= \Pr[\mathrm{src}(R) \notin S \wedge \mathrm{h}_1(R) \cap S \neq \emptyset] \\ &= \frac{\sum_{v \in N_S \setminus S} p_{S,v}}{|V|}.\end{aligned} \tag{3}$$

Now only $\mathcal{E}_c$ is left. We estimate $\Pr[\mathcal{E}_c]$ by the RIS method. For $\mathcal{E}_c$ to happen, $\mathrm{h}_{2^*}(R)$ must be non-empty. By definition, there must exist two nodes $w, u \in V$ such that both edges (w, u) and $(u, \mathrm{src}(R))$ are live in the sample. We refer to such a path composed of 2 live edges as a 2-hop live path. We define an RR set as a 2-hop RR set if the stochastic BFS produces a 2-hop live path. Our algorithm generates 2-hop RR sets only to measure $\Pr[\mathcal{E}_c]$. To guarantee that the RIS estimation is unbiased, the 2-hop RR sets should be generated with probabilities proportional to those in the original sample space of RR sets.

We first define some notations to facilitate the presentation. For each node $v \in V$, we assume a fixed order on v's inverse neighbors $I_v = \{v_1, v_2, \ldots, v_{\ell_v}\}$, where $\ell_v = |I_v|$. Accordingly, we have a fixed order on the incoming edges to v: (v_1, v), $(v_2, v), \ldots, (v_{\ell_v}, v)$. Among these edges, let η_i^v denote the probability that the first i edges $(1 \leq i \leq \ell_v)$ are all blocked in the BFS. We have

$$\eta_i^v = \prod\nolimits_{k=1}^{i}(1 - p_{v_k,v}). \tag{4}$$

For notational convenience, define $\eta_0^v = 1$.

We start by looking at generating a 2-hop RR set from a given source node v. For each node $v_i \in I_v$, let $I_{v_i} = \{v_{i,1}, v_{i,2}, \ldots, v_{i,\ell_{v_i}}\}$ denote the order on v_i's inverse neighbors, where $\ell_{v_i} = |I_{v_i}|$. First, we derive the probability distribution of the first 2-hop live path produced in the BFS. Let $A_{i,j}^v$ be the event that the path from $v_{i,j}$ through v_i to v is the first 2-hop live path. As illustrated in Fig. 2, when $A_{i,j}^v$ happens,

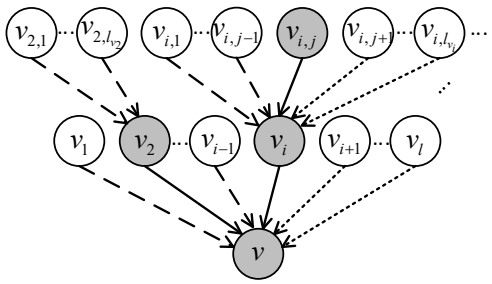


Fig. 2: Event $A_{i,j}^v\colon v_{i,j} \to v_i \to v$ is the first 2-hop live path. Dashed edges are blocked, solid edges are live and dotted edges are yet to be checked in the BFS.

(i) the edges $(v_{i,j}, v_i)$ and (v_i, v) are both live, (ii) for each node v_k where $1 \leq k \leq i-1$, there is no 2-hop live path going through v_k, i.e., either the edge (v_k, v) is blocked or all the incoming edges to v_k are blocked, and (iii) all the edges $(v_{i,1}, v_i), (v_{i,2}, v_i) \ldots, (v_{i,j-1}, v_i)$ are blocked. The probability for (i) to happen is $p_{v_i,v} \cdot p_{v_{i,j},v_i}$. Let α_k^v denote the probability that there is no 2-hop live path going through v_k. We have $\alpha_k^v = 1 - p_{v_k,v} + p_{v_k,v} \cdot \eta_{\ell_{v_k}}^{v_k}$ and the probability for (ii) to happen is $\prod_{k=1}^{i-1} \alpha_k^v$. The probability for (iii) to happen is $\eta_{j-1}^{v_i}$. Since all the edges involved are distinct, the probability for $A_{i,j}^v$ to happen is given by

$$\Pr[A_{i,j}^v] = p_{v_i,v} \cdot p_{v_{i,j},v_i} \cdot \eta_{j-1}^{v_i} \cdot \prod\nolimits_{k=1}^{i-1} \alpha_k^v. \tag{5}$$

A naive implementation to construct 2-hop RR sets from a given source node v is to precompute $\Pr[A_{i,j}^v]$ for all possible (i, j) pairs where $1 \leq i \leq \ell_v$ and $1 \leq j \leq \ell_{v_i}$, and generate the first 2-hop live path according to the probability distribution. The space complexity of the probability distribution can be $O(|V|)$. Since the source node is randomly chosen from V in RIS, the total space complexity for all the source nodes would be $O(|V|^2)$ which is infeasible for large-scale OSNs.

3.1.2 An Efficient Implementation

An important insight to our algorithm design is that it is not necessary to precompute all the probabilities $\Pr[A_{i,j}^v]$. The computation and storage complexities can be greatly reduced by decomposing the selection of the first 2-hop live path into two steps that identify v_i and $v_{i,j}$ respectively.

First, consider the selection of v_i. Let A_i^v be the event that the first 2-hop live path goes through v_i. We have

$$\begin{aligned}\Pr[A_i^v] &= \sum\nolimits_{j=1}^{\ell_{v_i}} \Pr[A_{i,j}^v] \\ &= \sum\nolimits_{j=1}^{\ell_{v_i}} \left(p_{v_i,v} \cdot p_{v_{i,j},v_i} \cdot \eta_{j-1}^{v_i} \cdot \prod\nolimits_{k=1}^{i-1} \alpha_k^v\right) \\ &= p_{v_i,v} \cdot \prod\nolimits_{k=1}^{i-1} \alpha_k^v \cdot \sum\nolimits_{j=1}^{\ell_{v_i}} \left(\eta_{j-1}^{v_i} - \eta_j^{v_i}\right) \\ &= p_{v_i,v} \cdot \prod\nolimits_{k=1}^{i-1} \alpha_k^v \cdot \left(1 - \eta_{\ell_{v_i}}^{v_i}\right) \\ &= (1 - \alpha_i^v) \cdot \prod\nolimits_{k=1}^{i-1} \alpha_k^v.\end{aligned} \tag{6}$$

Note that $\Pr[A_i^v]$ is the probability in the original sample space of RR sets. To generate 2-hop RR sets only, we need to normalize it by the probability for an RR set to have at least one 2-hop live path. Let β_v be the probability that an RR set generated from source v has at least one 2-hop live path. We have

$$\beta_v = \sum_{i=1}^{\ell_v} \Pr[A_i^v] = \sum_{i=1}^{\ell_v} \left((1 - \alpha_i^v) \cdot \prod_{k=1}^{i-1} \alpha_k^v\right) = 1 - \prod_{i=1}^{\ell_v} \alpha_i^v.$$

Algorithm 1: Select First 2-Hop Live Path

Input: A source node v, graph G, and precomputed η's and ϕ's
Output: Nodes v_i and $v_{i,j}$

Generate a random number $r \in [0, 1]$;
Find the lowest index i s.t. $(1 - \phi_i^v)/(1 - \phi_{\ell_v}^v) \geq r$ via binary search;
$\bar{r} \leftarrow \left(r - \frac{1-\phi_{i-1}^v}{1-\phi_{\ell_v}^v}\right) / \frac{\phi_{i-1}^v - \phi_i^v}{1-\phi_{\ell_v}^v}$;
Find the lowest index j s.t. $(1 - \eta_j^{v_i})/(1 - \eta_{\ell_{v_i}}^{v_i}) \geq \bar{r}$ via binary search;
return v_i and $v_{i,j}$;

Therefore, the node v_i in the first 2-hop live path can be selected according to the probability distribution of $\frac{\Pr[A_i^v]}{\beta_v}$. The corresponding cumulative probability is given by

$$\sum_{t=1}^{i} \frac{\Pr[A_t^v]}{\beta_v} = \frac{\sum_{t=1}^{i} \left((1-\alpha_t^v) \cdot \prod_{k=1}^{t-1} \alpha_k^v\right)}{\beta_v} = \frac{1 - \prod_{k=1}^{i} \alpha_k^v}{1 - \prod_{k=1}^{\ell_v} \alpha_k^v}.$$

Thus, v_i can be selected by generating a random number $r \in [0, 1]$ and finding the lowest index i such that $\frac{1-\prod_{k=1}^{i} \alpha_k^v}{1-\prod_{k=1}^{\ell_v} \alpha_k^v} \geq r$ via a binary search.

Next, consider the selection of $v_{i,j}$ given that the first 2-hop live path goes through v_i. By definition, $v_{i,j}$ will be selected with probability $\frac{\Pr[A_{i,j}^v]}{\Pr[A_i^v]}$. It follows from (5) and (6) that

$$\frac{\Pr[A_{i,j}^v]}{\Pr[A_i^v]} = \frac{p_{v_i,v} \cdot p_{v_{i,j},v_i} \cdot \eta_{j-1}^{v_i} \cdot \prod_{k=1}^{i-1} \alpha_k^v}{(1-\alpha_i^v) \cdot \prod_{k=1}^{i-1} \alpha_k^v} = \frac{p_{v_i,v} \cdot (\eta_{j-1}^{v_i} - \eta_j^{v_i})}{p_{v_i,v} \cdot (1 - \eta_{\ell_{v_i}}^{v_i})} = \frac{\eta_{j-1}^{v_i} - \eta_j^{v_i}}{1 - \eta_{\ell_{v_i}}^{v_i}}.$$

Thus, the corresponding cumulative probability is

$$\sum_{t=1}^{j} \frac{\Pr[A_{i,t}^v]}{\Pr[A_i^v]} = \frac{\sum_{t=1}^{j} (\eta_{t-1}^{v_i} - \eta_t^{v_i})}{1 - \eta_{\ell_{v_i}}^{v_i}} = \frac{1 - \eta_j^{v_i}}{1 - \eta_{\ell_{v_i}}^{v_i}}.$$

Let $\phi_i^v = \prod_{k=1}^{i} \alpha_k^v$. Consider the random number r generated for selecting v_i. Recall that when v_i is selected, the random number r falls in the range of $(\frac{1-\phi_{i-1}^v}{1-\phi_{\ell_v}^v}, \frac{1-\phi_i^v}{1-\phi_{\ell_v}^v}]$. Then, the value of $\bar{r} = \left(r - \frac{1-\phi_{i-1}^v}{1-\phi_{\ell_v}^v}\right) / \frac{\phi_{i-1}^v - \phi_i^v}{1-\phi_{\ell_v}^v}$ falls in the range of $[0, 1]$ uniformly. Therefore, $v_{i,j}$ can be selected by finding the lowest index j such that $\frac{1-\eta_j^{v_i}}{1-\eta_{\ell_{v_i}}^{v_i}} \geq \bar{r}$ via a binary search. To implement the selections of v_i and $v_{i,j}$ efficiently, we can precompute all the values η_i^v for every node v and every $i \leq \ell_v$. We also precompute all the values $\phi_i^v = \prod_{k=1}^{i} \alpha_k^v$ for every node v and every $i \leq \ell_v$ by computing α_i^v based on $\eta_{\ell_{v_i}}^{v_i}$. Algorithm 1 shows the details of selecting the first 2-hop live path from source v.

Selecting Source Node. Note that the probability for an RR set generated from source v to be a 2-hop RR set is β_v, and the source node is selected uniformly at random in RIS. Thus, in order to generate 2-hop RR sets only while following the probability distribution in the original sample space, we select a source node v in 2-hop sampling with probability being proportional to β_v. That is, the probability of selecting v as source node is given by

$$\Pr[\text{src} = v] = \frac{\beta_v}{\kappa}, \text{ where } \kappa = \sum\nolimits_{v \in V} \beta_v. \tag{7}$$

Adding Nodes $\{v_1, v_2, \ldots, v_{i-1}\}$. Suppose that v_i is selected for the first 2-hop live path from the source node v. For each v's inverse neighbor v_k where $k < i$, there is no 2-hop live path going through v_k. As a result, two cases may happen: (i) edge (v_k, v) is blocked, or (ii) edge (v_k, v) is live and edges $(v_{k,1}, v_k), (v_{k,2}, v_k), \ldots, (v_{k,\ell_{v_k}}, v_k)$ are all blocked. The probability for (i) to happen is $1 - p_{v_k,v}$, and the probability for (ii) to happen is $p_{v_k,v} \cdot \eta_{\ell_{v_k}}^{v_k} = \alpha_k^v - (1 - p_{v_k,v})$. For case (i), node v_k should not be marked as traversed in the BFS because the incoming edges to v_k are possibly live and may give rise to new nodes to be added to the RR set if there is a live path (more than 1 hop) from v_k to the source node v. For case (ii), node v_k is added to the RR set and should be marked as traversed since no incoming edge to v_k is live. Thus, for each node v_k where $k < i$, we add it to the RR set and mark it as traversed with probability

$$\Pr[T_k] = \frac{\alpha_k^v - (1 - p_{v_k,v})}{\alpha_k^v} = \frac{p_{v_k,v} \cdot \eta_{\ell_{v_k}}^{v_k}}{1 - p_{v_k,v} + p_{v_k,v} \cdot \eta_{\ell_{v_k}}^{v_k}}. \tag{8}$$

Algorithm 2: 2-hop Sampling

Input: A graph $G = (V, E)$
Output: A 2-hop RR set $R = \langle \text{src}(R), \text{h}_1(R), \text{h}_{2^*}(R) \rangle$

Select $v \in V$ according to probability distribution in (7);
$\text{src}(R) \leftarrow v$, mark v as traversed;
Select the first 2-hop live path $v_{i,j} \rightarrow v_i \rightarrow v$ via Algorithm 1;
$\text{h}_1(R) \leftarrow \{v_i\}$, mark v_i as traversed;
foreach $k \leftarrow 1$ **to** $i - 1$ **do**
    With probability $\Pr[T_k]$ in (8) do: $\text{h}_1(R) \leftarrow \text{h}_1(R) \cup \{v_k\}$, mark v_k as traversed;
$\text{h}_{2^*}(R) \leftarrow \emptyset$;
foreach $k \leftarrow i + 1$ **to** ℓ_v **do**
    if v_k *is not traversed* **then**
        With probability $p_{v_k,v}$ do: $\text{h}_1(R) \leftarrow \text{h}_1(R) \cup \{v_k\}$, add v_k into Q, mark v_k as traversed;
if $v_{i,j}$ *is not traversed* **then**
    $\text{h}_{2^*}(R) \leftarrow \text{h}_{2^*}(R) \cup \{v_{i,j}\}$, add $v_{i,j}$ into Q, mark $v_{i,j}$ as traversed;
foreach $k \leftarrow j + 1$ **to** ℓ_{v_i} **do**
    if $v_{i,k}$ *is not traversed* **then**
        With probability $p_{v_{i,k},v_i}$ do: $\text{h}_{2^*}(R) \leftarrow \text{h}_{2^*}(R) \cup \{v_{i,k}\}$, add $v_{i,k}$ into Q, mark $v_{i,k}$ as traversed;
while Q *is not empty* **do**
    $v \leftarrow$ the first node in Q;
    foreach $u \in I_v$ **do**
        if u *is not traversed* **then**
            With probability $p_{u,v}$ do: $\text{h}_{2^*}(R) \leftarrow \text{h}_{2^*}(R) \cup \{u\}$, add u into Q, mark u as traversed;
return $R = \langle \text{src}(R), \text{h}_1(R), \text{h}_{2^*}(R) \rangle$;

Algorithm 2 summarizes the generation of a 2-hop RR set. The 2-hop RR set R returned by Algorithm 2 distinguishes between the nodes $\text{src}(R)$, $\text{h}_1(R)$ and $\text{h}_{2^*}(R)$. First, a node v is selected as the source with the probability distribution of $\Pr[\text{src} = v]$ according to (7) (line 1). Then, the first 2-hop live path $v_{i,j} \rightarrow v_i \rightarrow v$ is chosen using Algorithm 1 (line 3) and v_i is added to $\text{h}_1(R)$ (line 4). Next, each inverse neighbor v_k of v where $k < i$ is marked as traversed and added to $\text{h}_1(R)$ with probability $\Pr[T_k]$ (line 6). Each inverse neighbor v_k of v where $i < k \leq \ell_v$ is marked as traversed and added to $\text{h}_1(R)$ with probability $p_{v_k,v}$ (line 10). After that, $v_{i,j}$ is added to $\text{h}_{2^*}(R)$ if it is not traversed before (line 12). Note that $v_{i,j}$ can be added to $\text{h}_{2^*}(R)$ only after all the inverse neighbors of v are checked so

as to ensure that $v_{i,j}$ does not belong to $\mathrm{h}_1(R)$. The remaining sampling process follows normal RIS.

3.2 SkipEdge: A Fast Sampling Method

In a vanilla implementation of the RR set construction, we need to generate a random number for each edge encountered in the BFS to determine whether the edge is live or blocked. In this section, we develop a new algorithm, referred to as SkipEdge, to speed up the generation of 2-hop RR sets. Our SkipEdge algorithm attempts to skip some edges if they are likely to be blocked together.

Consider a node u encountered in the BFS. Let $\Pr[B_i^u]$ be the probability that (u_i, u) is the first live edge among all the incoming edges $(u_1, u), (u_2, u), \ldots, (u_{\ell_u}, u)$ to u. That is, edges $(u_1, u), (u_2, u), \ldots, (u_{i-1}, u)$ are all blocked and edge (u_i, u) is live. Then, $\Pr[B_i^u]$ is given by

$$\Pr[B_i^u] = \eta_{i-1}^u \cdot p_{u_i,u} = \eta_{i-1}^u - \eta_i^u.$$

Let ζ_0^u be the probability that there is at least one live edge among all the incoming edges to u. We have

$$\zeta_0^u = \sum\nolimits_{k=1}^{\ell_u} \Pr[B_k^u] = \sum\nolimits_{k=1}^{\ell_u} \left(\eta_{k-1}^u - \eta_k^u\right) = 1 - \eta_{\ell_u}^u. \quad (9)$$

Note that $\eta_{\ell_u}^u$ is the probability that all the incoming nodes to u are blocked. Therefore, with probability $1 - \zeta_0^u$, we can directly terminate the sampling process for the incoming edges to u. With the other ζ_0^u probability, we can select the first live edge according to the probability distribution of $\Pr[B_i^u]$ $(1 \le i \le \ell_u)$.

This idea can be extended to locate all the subsequent live edges since the probability distribution of each new live edge depends on only the last live edge identified. In general, let (u_i, u) be a live edge identified. Given that (u_i, u) is a live edge, let $\Pr[B_{j|i}^u]$ be the probability that (u_j, u) is the next live edge $(j > i)$. That is, edges $(u_{i+1}, u), (u_{i+2}, u), \ldots, (u_{j-1}, u)$ are all blocked and edge (u_j, u) is live. Then, $\Pr[B_{j|i}^u]$ is given by

$$\Pr[B_{j|i}^u] = \frac{\eta_{j-1}^u}{\eta_i^u} \cdot p_{u_j,u} = \frac{\eta_{j-1}^u - \eta_j^u}{\eta_i^u}.$$

Let ζ_i^u be the probability that there is at least one live edge from an inverse neighbor of u indexed higher than i. Similar to the derivation of (9), we have

$$\zeta_i^u = \sum\nolimits_{k=i+1}^{\ell_u} \Pr[B_{k|i}^u] = 1 - \frac{\eta_{\ell_u}^u}{\eta_i^u}.$$

Hence, with probability $1 - \zeta_i^u$, we can terminate the sampling process after finding the live edge (u_i, u). With the other ζ_i^u probability, we can select the next live edge according to the probability distribution of $\Pr[B_{j|i}^u]$ $(i < j \le \ell_u)$.

To make the selection efficient, we can use a binary search. In particular, the cumulative probability is given by

$$\sum\nolimits_{k=i+1}^{j} \Pr[B_{k|i}^u] = 1 - \frac{\eta_j^u}{\eta_i^u}.$$

Thus, the selection of the next live edge can be conducted by generating a random number $r \in [0, 1]$ and identifying the lowest index j such that $1 - \frac{\eta_j^u}{\eta_i^u} \ge r$ via a binary search. Note that all the values η_i^u are precomputed in our 2-hop sampling method. Algorithm 3 shows the details of the SkipEdge algorithm to compute the list of u's inverse neighbors to traverse when a node u is encountered in the BFS.

Algorithm 3: SkipEdge

Input: Node u's inverse neighbors $I_u = \{u_1, u_2, \ldots, u_{\ell_u}\}$ and the corresponding $(\eta_1^u, \eta_2^u, \ldots, \eta_{\ell_u}^u)$
Output: A list of u's inverse neighbors to traverse

1 Initialize $i \leftarrow 0$, $\mathrm{Act}_u \leftarrow \emptyset$;
2 **while** $i < \ell_u$ **do**
3 Generate a random number $r \in [0, 1]$;
4 **if** $r > 1 - \eta_{\ell_u}^u / \eta_i^u$ **then break**;
5 Find the lowest index j s.t. $1 - \eta_j^u / \eta_i^u \ge r$ via binary search;
6 Add u_j into Act_u;
7 Update $i \leftarrow j$;
8 **return** Act_u;

Complexity. The SkipEdge algorithm takes $\Theta(\log(\ell_u))$ time (binary search in line 5) to identify each live edge to u. Let E_u be the total number of live edges and $\mathbb{E}[E_u]$ be its expected value. We have $\mathbb{E}[E_u] = \mu_u$, where $\mu_u = \sum_{k=1}^{\ell_u} p_{u_k,u}$ is the total propagation probability of all the incoming edges to u. As a result, the expected time complexity of SkipEdge is $\Theta\big(\log(\ell_u) \cdot (1 + \mu_u)\big)$. Note that the time complexity of a vanilla implementation that flips a coin for each edge is $\Theta(\ell_u)$. Since it normally holds that $\mu_u \ll \ell_u$ (e.g., under the widely-used Weighted Cascade model [28], [39], $\mu_u = 1$), our SkipEdge algorithm can significantly improve the sampling efficiency.

2-hop+: Improved 2-hop by SkipEdge. The normal RIS parts of the 2-hop sampling method (lines 8–10, 13–15 and 18–20 of Algorithm 2) can be augmented with the SkipEdge algorithm. In addition, SkipEdge can be applied to adding the nodes from $\{v_1, v_2, \ldots, v_{i-1}\}$ to the RR set (lines 5–6 of Algorithm 2) by inputting $\tilde{\eta}_j^v = \prod_{k=1}^{j} \frac{1 - p_{v_k,v}}{\alpha_k^v}$. We shall refer to the 2-hop sampling method equipped with SkipEdge as 2-hop+.

Remark. The value of η_i^u can be 0 if there exists an edge (u_k, u) where $p_{u_k,u}$ equals 1. In this case, we can place all the inverse neighbors u_k where $p_{u_k,u} = 1$ at the end of the ordered set I_u and exclude them from the sampling process since the edge (u_k, u) is always live. In addition, when all the incoming edges to a node v have the same propagation probability p, our SkipEdge technique leverages the geometric distribution sampling [22] to identify the index i_1 of the first live edge using a random number $r \in [0, 1]$ by $i_1 = \lceil \log(r) / \log(1 - p) \rceil$ and then the process iterates from the index $i_1 + 1$ until the index exceeds ℓ_u. The time complexity to select all the live edges to a given node u is $\Theta(1 + \mu_u)$.

3.3 Influence Estimation by 2-hop RR sets

Let Ω_{RIS} be the sample space of the standard RIS method. We use R_{RIS} to denote a random RR set generated by the RIS method with respect to the probability distribution $\Pr[R_{\mathsf{RIS}} = R]$ for each $R \in \Omega_{\mathsf{RIS}}$. Similarly, let $\Omega_{\text{2-hop}}$ be the sample space of our 2-hop (or 2-hop+) sampling method. We use $R_{\text{2-hop}}$ to denote a random 2-hop RR set generated by our 2-hop sampling method with respect to the probability distribution $\Pr[R_{\text{2-hop}} = R]$ for each $R \in \Omega_{\text{2-hop}}$. Then, for any 2-hop RR set $R(v) \in \Omega_{\text{2-hop}}$ with source node v, the probability that RIS generates $R(v)$ under the given source node v equals β_v times the probability that the 2-hop method generates $R(v)$ under the given source node v, i.e.,

$$\begin{aligned}&\Pr[R_{\mathsf{RIS}} = R(v) \mid \mathrm{src}(R_{\mathsf{RIS}}) = v]\\ &= \beta_v \cdot \Pr[R_{\text{2-hop}} = R(v) \mid \mathrm{src}(R_{\text{2-hop}}) = v].\end{aligned}$$

In addition, the probability of selecting source node v satisfies that $\Pr[\mathrm{src}(R_{\mathsf{RIS}}) = v] = \frac{1}{|V|}$ and $\Pr[\mathrm{src}(R_{\mathsf{2\text{-}hop}}) = v] = \frac{\beta_v}{\kappa}$. Therefore, for any node $v \in V$ and any 2-hop RR set $R(v) \in \Omega_{\mathsf{2\text{-}hop}}$ with source node v, we have

$$\begin{aligned}
&\Pr[R_{\mathsf{RIS}} = R(v)] \\
&= \Pr[R_{\mathsf{RIS}} = R(v) \mid \mathrm{src}(R_{\mathsf{RIS}}) = v] \cdot \Pr[\mathrm{src}(R_{\mathsf{RIS}}) = v] \\
&= \Pr[R_{\mathsf{2\text{-}hop}} = R(v) \mid \mathrm{src}(R_{\mathsf{2\text{-}hop}}) = v] \cdot \beta_v \cdot \frac{1}{|V|} \\
&= \Pr[R_{\mathsf{2\text{-}hop}} = R(v) \mid \mathrm{src}(R_{\mathsf{2\text{-}hop}}) = v] \cdot \frac{\kappa \Pr[\mathrm{src}(R_{\mathsf{2\text{-}hop}}) = v]}{|V|} \\
&= \Pr[R_{\mathsf{2\text{-}hop}} = R(v)] \cdot \frac{\kappa}{|V|}. \qquad (10)
\end{aligned}$$

For a random RR set R_{RIS} generated by the standard RIS method, recall that only $\mathcal{E}_a$, $\mathcal{E}_b$, and $\mathcal{E}_c$ can happen when $R_{\mathsf{RIS}} \cap S \neq \emptyset$. By (2) and (3) shown earlier, we have $\Pr[\mathcal{E}_a] = \frac{|S|}{|V|}$ and $\Pr[\mathcal{E}_b] = \frac{1}{|V|} \cdot \sum_{v \in N_S \setminus S} p_{S,v}$. Now let us come back to the estimation of $\Pr[\mathcal{E}_c]$ in (1).

Let Ω_c denote the set of all possible RR sets making event $\mathcal{E}_c$ happen. Recall that when event $\mathcal{E}_c$ happens, R_{RIS} must be a 2-hop RR set, which indicates that $\Omega_c \subseteq \Omega_{\mathsf{2\text{-}hop}}$, Thus, according to (10), we have

$$\Pr[\mathcal{E}_c] = \sum_{R \in \Omega_c} \Pr[R_{\mathsf{RIS}} = R] = \sum_{R \in \Omega_c} \Big(\Pr[R_{\mathsf{2\text{-}hop}} = R] \cdot \frac{\kappa}{|V|} \Big).$$

For a random 2-hop RR set $R_{\mathsf{2\text{-}hop}}$ returned by Algorithm 2, let us define a subset $R_{\mathsf{2\text{-}hop}^*} \subseteq R_{\mathsf{2\text{-}hop}}$ as follows: $R_{\mathsf{2\text{-}hop}^*} = \mathrm{h}_{2^*}(R_{\mathsf{2\text{-}hop}})$ if $\mathrm{src}(R_{\mathsf{2\text{-}hop}}) \notin S$ and $\mathrm{h}_1(R_{\mathsf{2\text{-}hop}}) \cap S = \emptyset$, and $R_{\mathsf{2\text{-}hop}^*} = \emptyset$ otherwise. Then, $\Pr[\mathcal{E}_c]$ can be rewritten as

$$\Pr[\mathcal{E}_c] = \Pr[R_{\mathsf{2\text{-}hop}^*} \cap S \neq \emptyset] \cdot \frac{\kappa}{|V|}.$$

Together with (1), (2) and (3) shown earlier, we can establish the relationship between the set $R_{\mathsf{2\text{-}hop}^*} \subseteq R_{\mathsf{2\text{-}hop}}$ and the influence spread $\sigma(S)$ as follows.

Theorem 1. *Given a seed set S and the subset $R_{\mathsf{2\text{-}hop}^*}$ of a random 2-hop RR set $R_{\mathsf{2\text{-}hop}}$ returned by Algorithm 2, we have*

$$\sigma(S) = |S| + \sum\nolimits_{v \in N_S \setminus S} p_{S,v} + \Pr[R_{\mathsf{2\text{-}hop}^*} \cap S \neq \emptyset] \cdot \kappa.$$

Influence Estimation. For a random 2-hop RR set $R_{\mathsf{2\text{-}hop}}$ generated by the 2-hop method, let X be a random variable defined as

$$X = \begin{cases} 1 & \text{if } R_{\mathsf{2\text{-}hop}^*} \cap S \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

By Theorem 1, we can get that

$$\mathbb{E}[X] = \Pr[R_{\mathsf{2\text{-}hop}^*} \cap S \neq \emptyset] = \frac{\sigma(S) - |S| - \sum_{v \in N_S \setminus S} p_{S,v}}{\kappa}.$$

Let $Z = X \cdot \frac{\kappa}{|V|} + \frac{|S| + \sum_{v \in N_S \setminus S} p_{S,v}}{|V|}$. Then, we have

$$\mathbb{E}[Z] = \mathbb{E}[X] \cdot \frac{\kappa}{|V|} + \frac{|S| + \sum_{v \in N_S \setminus S} p_{S,v}}{|V|} = \frac{\sigma(S)}{|V|}.$$

Consequently, the influence spread $\sigma(S)$ can be obtained by estimating $\mathbb{E}[Z]$ and scaling it up by a factor of $|V|$. Note that Z is a random variable bounded by $0 \leq a \leq Z \leq b$, where $a = \frac{|S| + \sum_{v \in N_S \setminus S} p_{S,v}}{|V|}$ and $b = a + \frac{\kappa}{|V|}$.

In practice, we can estimate $\mathbb{E}[X]$ or $\mathbb{E}[Z]$ by generating a sequence of random 2-hop RR sets. Specifically, let $\mathcal{R}_{\mathsf{2\text{-}hop}}$ be a set of 2-hop RR sets. Let $\Lambda(\mathcal{R}_{\mathsf{2\text{-}hop}^*}, S)$ be the number of $R_{\mathsf{2\text{-}hop}^*}$ samples in $\mathcal{R}_{\mathsf{2\text{-}hop}}$ that overlap S. Then, we can estimate $\mathbb{E}[X]$ by $\frac{\Lambda(\mathcal{R}_{\mathsf{2\text{-}hop}^*}, S)}{|\mathcal{R}_{\mathsf{2\text{-}hop}}|}$. Thus, the influence spread $\sigma(S)$ can be estimated by

$$\tilde{\sigma}(S) = |S| + \sum_{v \in N_S \setminus S} p_{S,v} + \frac{\Lambda(\mathcal{R}_{\mathsf{2\text{-}hop}^*}, S)}{|\mathcal{R}_{\mathsf{2\text{-}hop}}|} \cdot \kappa.$$

3.4 Complexity Analysis of 2-hop+

In an online social network, the number of edges is normally much larger than the number of nodes. We focus on the case when $|E| > |V|$ in the complexity analysis. For our 2-hop+ sampling method, the space complexity to store η_k^v, ϕ_k^v and $\Pr[T_k]$ $(1 \leq k \leq \ell_v)$ for a node v is $O(\ell_v)$ and the total space complexity is thus $O(|E|)$. The expected time complexity for generating a 2-hop RR set by our 2-hop+ sampling method is given as follows. Due to space limitations, details of all the missing proofs in the analysis are given in Appendix A of the supplementary file.

Lemma 2. *2-hop+ sampling takes $O\big(\frac{\mathrm{TP}}{\kappa} \cdot \mathbb{E}[\sigma_2(\{v^\diamond\})]\big)$ time in expectation to generate a 2-hop RR set, where $\mathrm{TP} = \sum_{v \in V} \big(\log(\ell_v) \cdot (1 + \mu_v)\big)$, $\sigma_2(\{v\}) = \Pr[R_{\mathsf{2\text{-}hop}} \cap \{v\} \neq \emptyset] \cdot \kappa$ and the expectation is over the randomness of $v^\diamond$ being chosen from V with probability $\Pr[v^\diamond = v] = \frac{\log(\ell_v) \cdot (1 + \mu_v)}{\mathrm{TP}}$.*

Taking the weighted cascade (WC) model as an example where $\sum_{k=1}^{\ell_v} p_{v_k,v} = 1$, we have $\mathrm{TP} = 2 \sum_{v \in V} \log(\ell_v) \leq 2|V| \cdot \log(\frac{|E|}{|V|})$ where the maximal is achieved when $\ell_v = \frac{|E|}{|V|}$ for each node $v \in V$. Hence, under the WC model, the expected time complexity is $O\big(\frac{|V| \cdot \log(\frac{|E|}{|V|})}{\kappa} \cdot \mathbb{E}[\sigma_2(\{v^\diamond\})]\big)$.

Comparison with Previous Work. Sadeh et al. [31] proposed a sampling method on the basis of Monte Carlo (MC) simulations. MC takes $O(E)$ space to record the graph. In a MC simulation, if a node v is activated, all its neighbors will be examined. Denote by $\Pr[S \to v]$ the probability that S activates v and by d_v the out-degree of v. Then, the expected time complexity of MC is $O\big(\sum_{v \in V} \big(\Pr[S \to v] \cdot d_v\big)\big) = O\big(|E| \cdot \mathbb{E}[\Pr[S \to v^\triangle]]\big)$, where the expectation is taken over the randomness of $v^\triangle$ being chosen from V with probability proportional to its out-degree, i.e., $\Pr[v^\triangle = v] = \frac{d_v}{|E|}$.

In addition, the standard RIS method [4] takes $O(E)$ space to record the graph and $O\big(\frac{|E|}{|V|} \cdot \mathbb{E}[\sigma(\{v^*\})]\big)$ time in expectation to generate a random RR set [39], where v^* is a random node chosen from V with probability proportional to its in-degree, i.e., $\Pr[v^* = v] = \frac{\ell_v}{|E|}$.

We note that Nguyen et al. [28] proposed an importance influence sampling method, called SKIS, to generate only the non-singular RR sets which contain at least one node other than the source. Such a random non-singular RR set R_{SKIS} satisfies that

$$\sigma(S) = \Pr[R_{\mathsf{SKIS}} \cap S \neq \emptyset] \cdot \Gamma + \sum\nolimits_{v \in S} (1 - \gamma_v),$$

where γ_v is the probability for the source v to have a non-singular sample and $\Gamma = \sum_{v \in V} \gamma_v$. SKIS requires to precompute and store all the values η_k^v $(1 \leq k \leq \ell_v)$, which uses a total of $O(|E|)$ space. Let $\sigma_1(\{v\}) = \Pr[R_{\mathsf{SKIS}} \cap \{v\} \neq \emptyset] \cdot \Gamma$. Similar to our analysis for 2-hop+, the expected time complexity to generate an SKIS sample is $O\big(\frac{|E|}{\Gamma} \cdot \mathbb{E}[\sigma_1(\{v^*\})]\big)$ where v^* is a random node chosen from V with probability proportional to its in-degree, i.e., $\Pr[v^* = v] = \frac{\ell_v}{|E|}$.

TABLE 1: Complexity Results under General Cases.

Method	Space	Time
MC [31]	$O(\|E\|)$	$O\big(\|E\| \cdot \mathbb{E}[\Pr[S \to v^{\triangle}]]\big)$
RIS [4]	$O(\|E\|)$	$O\big(\frac{\|E\|}{\|V\|} \cdot \mathbb{E}[\sigma(\{v^{\star}\})]\big)$
SKIS [28]	$O(\|E\|)$	$O\big(\frac{\|E\|}{\Gamma} \cdot \mathbb{E}[\sigma_1(\{v^{\star}\})]\big)$
SUBSIM [16]	$O(\|E\|)$	$O\big(\frac{\mathrm{ITP}}{\|V\|} \cdot \mathbb{E}[\sigma(\{v^{\circ}\})]\big)$
2-hop+	$O(\|E\|)$	$O\big(\frac{\mathrm{TP}}{\kappa} \cdot \mathbb{E}[\sigma_2(\{v^{\diamond}\})]\big)$

Very recently, Guo et al. [16] proposed a sampling technique, called SUBSIM, to improve the sampling efficiency of RIS. An index-free method is adopted by SUBSIM to sample the live edges to a node. Specifically, for a node v, the probabilities of the incoming edges $\{p_1, p_2, \ldots, p_{\ell_v}\}$ are first sorted in descending order. Then, those in the index range of $[2^k, 2^{k+1}]$ are put into a bucket b_k with $k = 0, 1, \ldots, \lfloor \log_2 \ell_v \rfloor$. In each bucket b_k, p_{2^k} is used as the probability for the geometric distribution $G(p_{2^k})$. When the index i sampled from $G(p_{2^k})$ is greater than 2^k, no live edge is sampled from b_k. Otherwise, it switches to position $2^k + i$ and sample the incoming neighbor v_{2^k+i} with probability p_{2^k+i}/p_{2^k}. The time complexity for this technique to sample the incoming edges to v is $O(1 + \mu_v + \log(\ell_v))$. Let $\mathrm{ITP} = \sum_{v \in V} \big(1 + \mu_v + \log(\ell_v)\big)$. Similar to our analysis for 2-hop+, the expected time complexity to generate an RIS sample by SUBSIM is $O\big(\frac{\mathrm{ITP}}{|V|} \cdot \mathbb{E}[\sigma(\{v^{\circ}\})]\big)$ where v° is a random node chosen from V with probability $\Pr[v^{\circ} = v] = \frac{1+\mu_v+\log(\ell_v)}{\mathrm{ITP}}$. Note that in the special case when all the incoming edges to a node v have the same propagation probability p, SUBSIM leverages the geometric distribution sampling [22] to identify each live edge using a random number $r \in [0, 1]$ by computing $\lceil \log(r)/\log(1-p) \rceil$. Let $\mathrm{STP} = \sum_{v \in V}(1 + \mu_v)$. In such a special case, the expected time complexity of SUBSIM to generate an RIS sample is $O\big(\frac{\mathrm{STP}}{|V|} \cdot \mathbb{E}[\sigma(\{v^{\star}\})]\big)$, where $v^{\star}$ is a random node chosen from V with probability $\Pr[v^{\star} = v] = \frac{1+\mu_v}{\mathrm{STP}}$. We also use the geometric distribution sampling [22] in our method when all the incoming edges to a node share the same probability. In particular, we adopt the geometric distribution to identify the live edges for lines 8–10, 13–15 and 18–20 of Algorithm 2 and adopt our general SkipEdge for lines 5–6 of Algorithm 2 since the probabilities $\Pr[T_k]$ $(1 \le k < i)$ are not uniform. As a result, the expected time complexity to generate a 2-hop sample by 2-hop+ is between $O\big(\frac{\mathrm{STP}}{\kappa} \cdot \mathbb{E}[\sigma_2(\{v^{\star}\})]\big)$ and $O\big(\frac{\mathrm{TP}}{\kappa} \cdot \mathbb{E}[\sigma_2(\{v^{\diamond}\})]\big)$.

For ease of reference, Table 1 summarizes the complexity results of algorithms for general cases, including the existing MC [31], RIS [4], SKIS [28] and SUBSIM [16] algorithms, and our 2-hop+ algorithm.

3.5 Influence Maximization by 2-hop RR Sets

Influence maximization [20] is one of the most important applications on the basis of influence estimation. Leveraging the monotonicity and submodularity of influence spread [20], the standard greedy algorithm is used for seed selection that can achieve an approximation ratio of $1 - 1/\mathrm{e}$ [26]. However, we find that the relation between the influence spread and a random 2-hop+ RR set given in Theorem 1 may not suitable for influence maximization, since the term $\Pr[R_{\text{2-hop}^{\star}} \cap S \neq \emptyset]$ is no longer monotone and submodular. To tackle the application of influence maximization, we establish a new relation as follows.

Theorem 2. *For notational convenience, let $p_{u,u} = 1$ for every node $u \in V$. Moreover, let $\alpha_{u,v} = 1 - p_{u,v} + p_{u,v} \cdot \eta^u_{\ell_u}$. Given a seed set S, for a random RR set $R_{\text{2-hop}}$ returned by Algorithm 2, we have*

$$\sigma(S) = \sum_{v \in N_S \cup S} \Big((1 - \beta_v) \cdot \big(1 - \prod_{u \in (I_v \cup \{v\}) \cap S} \frac{1 - p_{u,v}}{\alpha_{u,v}}\big)\Big) + \Pr[R_{\text{2-hop}} \cap S \neq \emptyset] \cdot \kappa. \tag{11}$$

Note that both two terms in the right hand side of (11) are monotone and submodular which indicates that the greedy algorithm can be applied to tackle the influence maximization problem with approximation ratio of $1 - 1/\mathrm{e}$. Specifically, let $\tilde{\sigma}(S)$ be an estimate of $\sigma(S)$ by generating a set $\mathcal{R}_{\text{2-hop}}$ of 2-hop RR sets using Algorithm 2, which replaces $\Pr[R_{\text{2-hop}} \cap S \neq \emptyset]$ with $\frac{\Lambda(\mathcal{R}_{\text{2-hop}}, S)}{|\mathcal{R}_{\text{2-hop}}|}$ in (11), where $\Lambda(R_{\text{2-hop}}, S)$ is the number of 2-hop RR sets in $\mathcal{R}_{\text{2-hop}}$ that overlap S. We can see that the two terms in the right hand side are also monotone and submodular. Consequently, we can apply our 2-hop sampling method to the state-of-the-art OPIM-C framework [34] to address the influence maximization problem that can provide an approximation ratio of $(1 - 1/\mathrm{e} - \varepsilon)$ with high probability efficiently.

4 Improved Stopping Rule Algorithm

In this section, we propose an improved stopping rule algorithm based on the original stopping rule algorithm in [11] to estimate the mean of random variables falling in a given range of $[a, b]$ $(0 \le a < b)$ using the martingale-based concentration bounds. Our algorithm can be used to estimate the influence spread or $\mathbb{E}[Z]$ (discussed earlier) with (ε, δ)-approximation.

4.1 Martingale Concentration Bounds

Definition 1 ([9])**.** *A sequence of random variables $Y_1, Y_2, \ldots$ is a martingale if and only if $\mathbb{E}[Y_k \mid Y_1, Y_2, \ldots, Y_{k-1}] = Y_{k-1}$ and $\mathbb{E}[|Y_k|] < \infty$ for any k.*

Lemma 3 ([9])**.** *Let $M_1, M_2, \ldots, M_t$ be a martingale, such that $M_1 \le \alpha$, $M_k - M_{k-1} \le \alpha$ for any $2 \le k \le t$, and $\mathrm{Var}[M_1] + \sum_{k=2}^{t} \mathrm{Var}[M_k \mid M_1, M_2, \ldots, M_{k-1}] \le \beta$. Then, for any $\eta > 0$,*

$$\Pr[M_t - \mathbb{E}[M_t] \ge \eta] \le \exp\Big(-\frac{\eta^2}{\frac{2}{3}\alpha\eta + 2\beta}\Big).$$

Let $X_1, X_2, \ldots$ be a sequence of random variables satisfying: for each $k \ge 1$, $0 \le X_k \le 1$ and $\mathbb{E}[X_k \mid X_1, X_2, \ldots, X_{k-1}] = \mu_X$ where μ_X is a constant. Applying Lemma 3, we have the following result.

Lemma 4. *Given a fixed real number θ, let $\bar{X} = \frac{1}{\theta}\big(\sum_{k=1}^{\lfloor\theta\rfloor} X_k + (\theta - \lfloor\theta\rfloor) X_{\lceil\theta\rceil}\big)$. Then, for any $\lambda > 0$,*

$$\Pr[\bar{X} \ge \mu_X + \lambda] \le \exp\Big(-\frac{\lambda^2\theta}{2\mu_X + \frac{2}{3}\lambda}\Big), \tag{12}$$

$$\Pr[\bar{X} \le \mu_X - \lambda] \le \exp\Big(-\frac{\lambda^2\theta}{2\mu_X}\Big). \tag{13}$$

To our knowledge, almost all the previous work [29, 40, 43] gives similar concentration bounds that require θ to be an integer. Interestingly, via a non-trivial analysis, we show that these concentration bounds also hold even when θ is a real number, which generalizes existing results. Lemma 4 is the core to devising our improved stopping rule algorithm with (ε, δ)-approximation that requires less samples.

Algorithm 4: Generalized Stopping Rule Algorithm

Input: Random variables $Z_1, Z_2, \ldots$ and $0 < \varepsilon, \delta < 1$
Output: An (ε, δ)-estimate $\tilde{\mu}_Z$ of μ_Z

1 $\Upsilon \leftarrow 2(b-a)(1+\varepsilon)(\frac{b-a}{b} + \frac{1}{3}\varepsilon)\ln(\frac{2}{\delta})\frac{1}{\varepsilon^2}$;
2 Initialize $\theta \leftarrow 0$, $\Sigma \leftarrow 0$;
3 **while** $\Sigma < \Upsilon$ **do**
4 $\quad \Sigma \leftarrow \Sigma + Z_\theta$, $\theta \leftarrow \theta + 1$;
5 $\theta \leftarrow \theta - \frac{\Sigma - \Upsilon}{Z_\theta}$;
6 **return** $\tilde{\mu}_Z = \frac{\Upsilon}{\theta}$;

4.2 Improved Stopping Rule Algorithm

Based on Lemma 4, we give an improved stopping rule algorithm (Algorithm 4) that yields an (ε, δ)-estimate of the mean μ_Z of random variables Z_k satisfying: for each $k \geq 1$, (i) $a \leq Z_k \leq b$ where $0 \leq a < b$, and (ii) $\mathbb{E}[Z_{k+1} \mid Z_1, Z_2, \ldots, Z_k] = \mu_Z$. Algorithm 4 first computes a threshold Υ that guards the stopping time on the basis of the bounds a and b of random variables and the input accuracy parameters ε and δ (Line 1). Then, it accumulates the observed values of random samples until the accumulation Σ reaches the threshold Υ (Lines 3–4). Finally, it slightly adjusts the total number of samples θ to $\theta - \frac{\Sigma - \Upsilon}{Z_\theta}$ and returns an estimate $\frac{\Upsilon}{\theta}$ (Line 5–6), where the adjustment of sample size is a key step to ensure the accuracy of influence estimation leveraging Lemma 4.

Theorem 3. *Algorithm 4 returns an (ε, δ)-estimate $\tilde{\mu}_Z$ of μ_Z, i.e.,*

$$\Pr[(1-\varepsilon)\mu_Z \leq \tilde{\mu}_Z \leq (1+\varepsilon)\mu_Z] \geq 1 - \delta, \tag{14}$$

and the number of samples θ satisfies

$$\Pr\Big[\theta \leq \frac{\Upsilon}{(1-\varepsilon)\mu_Z}\Big] \geq 1 - \frac{\delta}{2}. \tag{15}$$

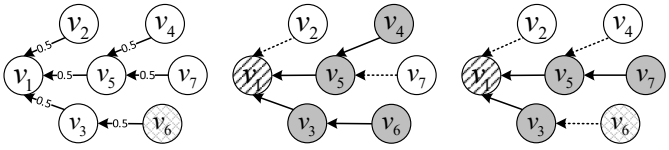
Comparison with Previous Work [29]. Nguyen et al. [29] also proposed a stopping rule algorithm to estimate the mean of random variables in a given range $[a, b]$. The setting of Υ by Nguyen et al. [29] is

$$\Upsilon_{\mathrm{N}} = 2(b-a)(1+\varepsilon)(1+\frac{1}{3}\varepsilon')\ln(\frac{2}{\delta})\frac{1}{\varepsilon'^2},$$

where $\varepsilon' = \varepsilon(1 - \frac{\varepsilon b}{(2+\frac{2}{3}\varepsilon)\ln(\frac{2}{\delta})(b-a)}) < \varepsilon$. It is easy to see that Υ_{N} is not only rather complicated but also larger than our setting of Υ, which indicates that our algorithm always requires less number of samples than Nguyen et al. [29]. We argue that the key reason behind is that our Lemma 4 no longer limits θ to integers only and θ can be any real number. This property allows our algorithm (Algorithm 4) to ensure that the total value of all observed random variables equals to a fixed constant Υ, i.e., $\sum_{k=1}^{\lfloor\theta\rfloor} Z_k + (\theta - \lfloor\theta\rfloor)Z_{\lceil\theta\rceil} = \Upsilon$ (Line 5). In contrast, Nguyen et al.'s algorithm [29] utilizes concentration bounds that require θ to be an integer so that $\sum_{k=1}^{\theta} Z_k$ is in the range of $[\Upsilon_{\mathrm{N}}, \Upsilon_{\mathrm{N}} + b)$. Such an uncertainty requires an extra relation between b and Υ_{N} to ensure the estimation accuracy, resulting in more samples than our algorithm.

4.3 (ε, δ)-Approximate Influence Estimation

Recall from Section 3 that by the 2-hop+ sampling method, influence estimation degenerates to estimating the mean μ_Z of a sequence of random variables $Z_1, Z_2, \ldots$, where $a \leq Z_k \leq b$, $a = \frac{|S| + \sum_{v \in N_S \setminus S} p_{S,v}}{|V|}$ and $b = a + \frac{\kappa}{|V|}$. Since each 2-hop RR set is generated independently, for each $k \geq 1$, we have



(a) An Example Graph (b) A 2-hop Sample R (c) A 2-hop Sample R'

Fig. 3: A social graph and 2 possible 2-hop samples. Dashed arrows indicate blocked edges. Seed set is $\{v_6\}$.

$\mathbb{E}[Z_k \mid Z_1, Z_2, \ldots, Z_{k-1}] = \mu_Z = \frac{\sigma(S)}{|V|}$. Based on Theorem 3, an (ε, δ)-estimate $\tilde{\mu}_Z$ of μ_Z can be obtained by using Algorithm 4 and then an (ε, δ)-estimate of the influence spread can be obtained by $\tilde{\sigma}(S) = |V| \cdot \tilde{\mu}_Z$.

Example 2. *Fig. 3 shows an example social graph and 2 possible 2-hop samples. The graph in Fig. 3(a) has 7 nodes and 6 edges where each edge is associated with a probability of 0.5. In this example, there are three 2-hop paths, i.e., $v_4 \to v_5 \to v_1$, $v_7 \to v_5 \to v_1$ and $v_6 \to v_3 \to v_1$. Thus, by definition, only v_1 can be selected as the source node for generating 2-hop samples, since there is no 2-hop path ending at other nodes than v_1. That is, $\beta_{v_k} = 0$ for $k \geq 2$. Meanwhile, we can compute that $\kappa = \beta_{v_1} = 1 - \alpha_{v_3}^{v_1} \cdot \alpha_{v_5}^{v_1} = \frac{17}{32}$ where $\alpha_{v_3}^{v_1} = 1 - p_{v_3,v_1}p_{v_6,v_3} = \frac{3}{4}$ and $\alpha_{v_5}^{v_1} = 1 - p_{v_5,v_1}(1-(1-p_{v_4,v_5})(1-p_{v_7,v_5})) = \frac{5}{8}$ are the probabilities of no 2-hop path going through v_3 and v_5, respectively. Consider a seed set $S = \{v_6\}$. For the RR set R in Fig. 3(b), we have $\mathrm{src}(R) = v_1$, $\mathrm{h}_1(R) = \{v_3, v_5\}$ and $\mathrm{h}_{2^*}(R) = \{v_4, v_6\}$, while for the RR set R' in Fig. 3(c), $\mathrm{src}(R') = v_1$, $\mathrm{h}_1(R') = \{v_3, v_5\}$ and $\mathrm{h}_{2^*}(R') = \{v_7\}$. This indicates that $R_{\text{2-hop}^*} \cap S \neq \emptyset$ and $R'_{\text{2-hop}^*} \cap S = \emptyset$. According to Algorithm 4 (Line 4), if an RR set R in Fig. 3(b) is generated, Σ is increased by $Z = b$, whereas if an RR set R in Fig. 3(c) is generated, Σ is increased by $Z = a$, where $a = \frac{|S| + \sum_{v \in N_S \setminus S} p_{S,v}}{|V|} = \frac{3}{14}$ and $b = a + \frac{\kappa}{|V|} = \frac{65}{224}$. Given $\varepsilon = 0.01$ and $\delta = 0.001$, Algorithm 4 first gets the threshold $\Upsilon = 3086.41$ (Line 1). Then, it keeps generating 2-hop samples until Σ reaches the threshold Υ (Lines 3–4). Suppose that a total of 12385 RR sets are generated with the accumulation $\Sigma = 3086.67$ and the last RR set has $Z_\theta = 0.29$. Then, it slightly adjusts the total number of samples to 12384.1 and obtains $\tilde{\mu}_Z = 0.2492$ (Line 5–6). As a result, by Theorem 1, we can get an estimate of $\sigma(\{v_6\})$ as $\tilde{\sigma}(\{v_6\}) = 1.745$. On the other hand, it is easy to verify that $\sigma(\{v_6\}) = 1.75$ such that the estimate via Algorithm 4 is clearly within a relative error of 0.01.*

Complexity Analysis. Sadeh et al. [31] used a fixed number of $\tau\varepsilon^{-2}\delta^{-1}$ Monte Carlo simulations to estimate the expected influence spread $\sigma(S)$ with (ε, δ) guarantee under the assumption that the influence spread within τ hops can be a good estimate of the total influence spread where τ is the number of diffusion steps satisfying $1 \leq \tau \leq |V|$. Thus, the total time complexity is $O\big(|E| \cdot \frac{\tau}{\varepsilon^2 \cdot \delta} \cdot \mathbb{E}[\Pr[S \to v^{\Delta}]]\big)$. Meanwhile, the expected size of each sample is $O(\sum_{v \in V} \Pr[S \to v]) = O(|V| \cdot \mathbb{E}[\Pr[S \to \bar{v}]])$, where the expectation is over the randomness of $\bar{v}$ being uniformly chosen from V. Thus, the total expected size of samples is $\big(|V| \cdot \frac{\tau}{\varepsilon^2 \cdot \delta} \cdot \mathbb{E}[\Pr[S \to \bar{v}]]\big)$.

In addition, according to Wald's equation [41], our stopping rule algorithm generates an expected number of samples in the range of $[\frac{\Upsilon}{\mu_Z}, \frac{\Upsilon}{\mu_Z} + 1]$ and takes an expected time of $O\big(\mathrm{EPT} \cdot \frac{\Upsilon}{\mu_Z}\big) = O\big(\mathrm{EPT} \cdot \frac{\Upsilon \cdot |V|}{\sigma(S)}\big)$, where EPT is the expected time complexity for generating one sample. The value of Υ is

TABLE 2: Expected total sample size and time complexity for obtaining an (ε, δ)-estimate of the influence spread $\sigma(S)$.

Method	Total Sample Size	Time
MC [31]	$O(\frac{\lvert V\rvert\tau}{\varepsilon^2\delta} \cdot \mathbb{E}[\Pr[S \to \bar{v}]])$	$O(\frac{\lvert E\rvert\tau}{\varepsilon^2\delta} \cdot \mathbb{E}[\Pr[S \to v^{\triangle}]])$
RIS [4]	$O(\frac{\lvert V\rvert \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma(\{\bar{v}\})])$	$O(\frac{\lvert E\rvert \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma(\{v^*\})])$
SKIS [28]	$O(\frac{\Gamma \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma_1(\{\bar{v}\})])$	$O(\frac{\lvert E\rvert \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma_1(\{v^*\})])$
SUBSIM [16]	$O(\frac{\lvert V\rvert \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma(\{\bar{v}\})])$	$O(\frac{\mathrm{ITP} \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma(\{v^{\circ}\})])$
2-hop+	$O(\frac{\kappa \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma_2(\{\bar{v}\})])$	$O(\frac{\mathrm{TP} \ln(\frac{1}{\delta})}{\sigma(S)\varepsilon^2} \cdot \mathbb{E}[\sigma_2(\{v^{\diamond}\})])$

TABLE 3: Datasets.

Dataset	#nodes	#edges	Avg. degree	Type
Google+	107.6K	13.7M	254.1	Directed
LiveJournal	4.8M	69.0M	28.5	Directed
Orkut	3.1M	117.2M	76.3	Undirected
Twitter	41.7M	1.5G	70.5	Directed

$O(\frac{\kappa}{|V|} \cdot \frac{\ln(\frac{1}{\delta})}{\varepsilon^2})$, $O(\frac{\Gamma}{|V|} \cdot \frac{\ln(\frac{1}{\delta})}{\varepsilon^2})$ and $O(\frac{\ln(\frac{1}{\delta})}{\varepsilon^2})$ for 2-hop+, SKIS and RIS/SUBSIM respectively. Finally, together with EPT shown in Table 1, the expected time complexity to get an (ε, δ)-estimate of $\sigma(S)$ for different sampling methods is summarized in Table 2. Meanwhile, let EPS denote the expected size of one sample. Then, for RIS/SUBSIM, $\mathrm{EPS} = \mathbb{E}[|R|] = \mathbb{E}[\sigma(\{\bar{v}\})]$, where $\Pr[\bar{v} = v] = \frac{1}{|V|}$ for each $v \in V$. Similarly, for SKIS and 2-hop+, EPS is $\mathbb{E}[\sigma_1(\{\bar{v}\})]$ and $\mathbb{E}[\sigma_2(\{\bar{v}\})]$, respectively. As a result, combining with the expected number of samples generated gives the total expected size of samples for different algorithms as summarized in Table 2.

5 Experimental Evaluation

We evaluate the effectiveness and efficiency of our algorithms against the state-of-the-art solutions. All the experiments are run on a machine with Intel Xeon 2.4GHz CPU with 384GB memory.

5.1 Experimental Settings

Datasets. We use several real-world datasets [23, 24] with ranging from thousands to tens of millions of nodes. Table 3 shows the details of each dataset.

Parameter Settings. We focus on the IC model and adopt the following widely-used models to set the propagation probabilities.

- Weighted Cascade (WC) [28, 39]: The propagation probability $p_{u,v}$ of each edge (u, v) is set to the reciprocal of v's in-degree, i.e., $p_{u,v} = \frac{1}{|I_v|}$.
- Uniform (UNI) [20, 36]: The propagation probabilities of all the edges are set to the same value p. We set $p = 0.001$ in our experiments.
- Exponential distribution (EXP) [16]: The propagation probability of each edge is first randomly generated according to the probability density function $f(x) = \lambda e^{-\lambda x}$, where λ is set to 1. For each node v, the sum of the probabilities of its incoming edges is then scaled to 1.
- Weibull distribution (WEI) [16]: The propagation probability of each edge is first randomly generated according to the probability density function $f(x) = \frac{a}{b} \cdot (\frac{x}{b})^{a-1} e^{-(x/b)^a}$, where a and b are sampled from the range of [0,10] uniformly at random for each edge. For each node v, the sum of the probabilities of its incoming edges is then scaled to 1.

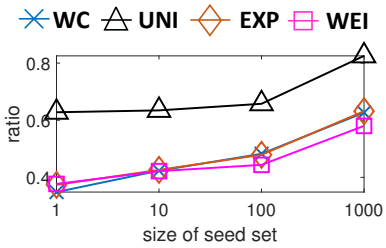


Fig. 4: Ratio of influence spread in 1 hop for Google+.

Algorithms. We compare our 2-hop+ method with the following four sampling methods.

- MC [31]. We generate $\varepsilon^{-2}\delta^{-1}$ Monte Carlo samples with $\tau = 1$ assuming that the total influence spread can be well approximated by the 1-hop influence.
- RIS [4]: The standard Reverse Influence Sampling method.
- SKIS [28]: The state-of-the-art reverse influence sampling method that generates non-singular samples only.
- SUBSIM [16]: The Subset Sampling method based on RIS.

To compare the efficiency of the methods, we implement all algorithms in C++ and equip the reverse sampling methods with our improved stopping rule algorithm (Section 4).

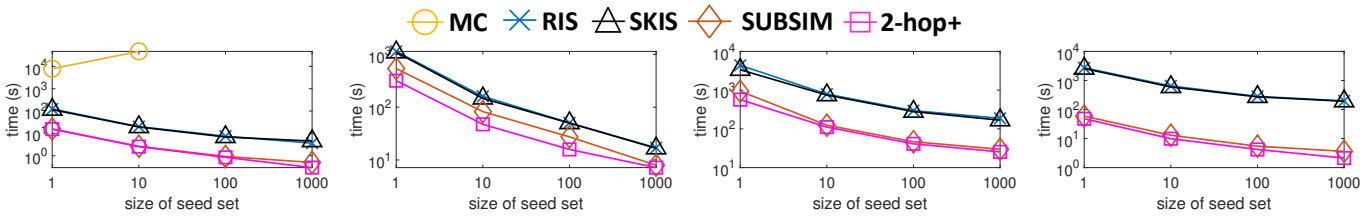
5.2 Experimental Results

5.2.1 *Efficiency of Influence Estimation*

We run our improved stopping rule algorithm to get an (ε, δ)-estimate of the influence spread for a given seed set. We experiment with different seed sets and observe similar performance trends since the trends of time complexities can hardly be affected by changing seed sets alone. Due to space limitations, we focus on reporting the results for the seed set that includes the top-k highest-degree nodes in the network. We test seed sets of size $k = 1$, 10, 100 and 1000. We conduct experiments for $\varepsilon = 0.01$ and $\delta = 0.001$.

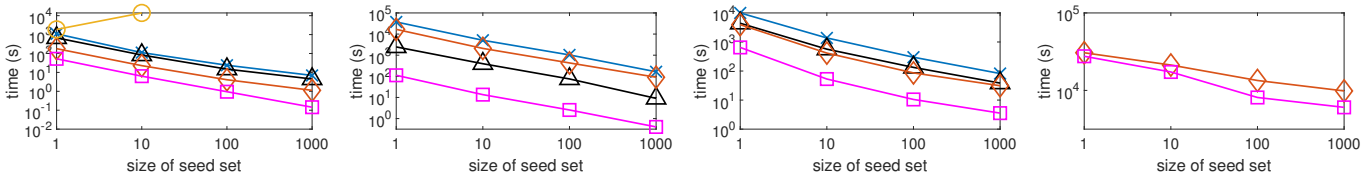
Running Time. Figs. 5–8 show the running time. We cap the running time at 24 hours and do not plot the results for unfinished methods (e.g., under the UNI model for the Twitter dataset as shown in Fig. 6, only 2-hop+ and SUBSIM methods can complete the estimation well within 24 hours whereas no other method can do it).

As shown in Figs. 5–8, the running time to generate $\varepsilon^{-2}\delta^{-1} = 10^7$ Monte Carlo simulations is significantly longer than the reverse sampling methods under various models. In fact, the execution can be finished within 24 hours only when both the seed size and the dataset are small, i.e., $|S| = 1$ or $|S| = 10$ on Google+. In addition, Fig. 4 plots the ratio of the 1-hop influence spread to the total influence spread for Google+. As can be seen, the 1-hop influence constitutes up to only around 80% under the UNIFORM model on Google+, which is far from reaching the approximation guarantee of $(0.01, 0.001)$. In other words, we need to further increase the value of τ until the influence spread within τ hops can be a good estimate of the total influence spread and thus more samples would be needed to achieve the approximation guarantee. However, as validated by Figs. 5–8, generating 10^7 samples already incurs much longer running time compared with other reverse sampling methods.



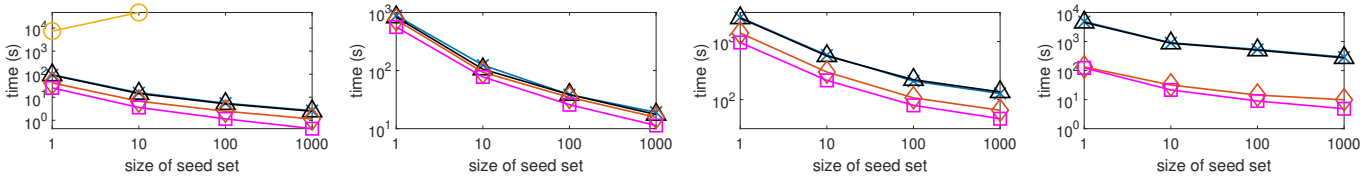
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 5: Running time for WC model.



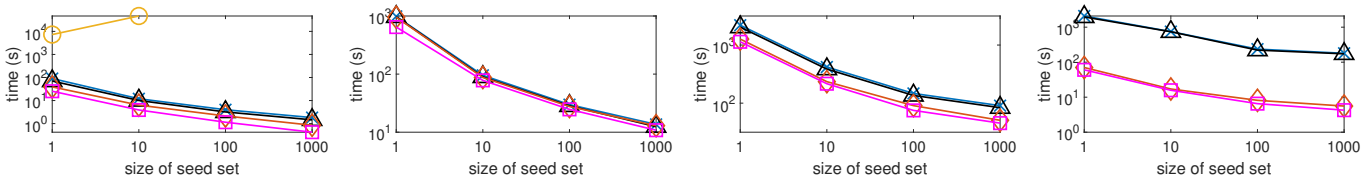
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 6: Running time for UNI model.



(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 7: Running time for EXP model.



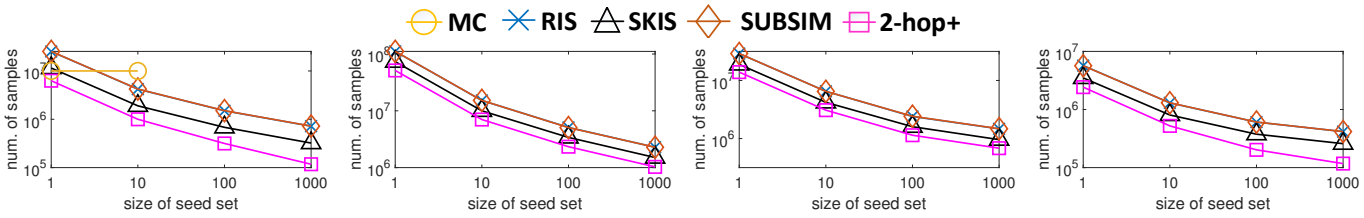
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 8: Running time for WEI model.

Figs. 5 and 6 show that our 2-hop+ method can always outperform the other baselines for both WC and UNI models. The speed-up is up to 90× times and 30× times for the WC and UNI models respectively compared with SKIS, and 92× times and 408× times for the WC and UNI models respectively compared with RIS. Compared with the state-of-the-art SUBSIM, our 2-hop+ method can achieve the speed-up up to 1.7× times and 224× times under the WC and UNI models respectively. In particular, our 2-hop+ method achieves the most notable speed-up for the LiveJournal dataset under the UNI model. This is because the propagation probability is relatively low under the UNI model and LiveJournal is more sparse compared to other datasets and thus we can avoid generating a greater number of samples with only 1-hop paths to improve the efficiency. Figs. 7 and 8 show that for the EXP and WEI models, our 2-hop+ method is still the best one with minimal running time to reach (ε, δ)-estimate. The speed-up is up to 56× times and 40× times for the EXP and WEI models respectively compared with SKIS, and 60× times and 48× times for the EXP and WEI models respectively compared with RIS. Compared with the state-of-the-art SUBSIM, our 2-hop+ method can achieve the speed-up up to 2.6× times and 2× times under the EXP and WEI models respectively.

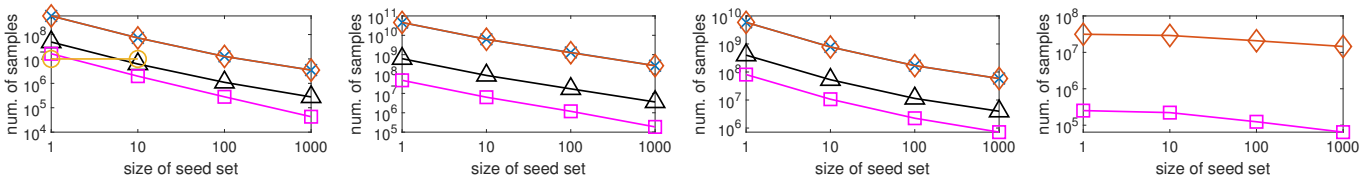
In summary, the experimental results validate that our proposed 2-hop+ method can significantly boost the efficiency of influence estimation.

Sample Size. Figs. 9–12 show the number of samples generated by different algorithms to reach (ε, δ)-estimate under the WC, UNI, EXP and WEI models respectively, and Figs. 13–16 show the corresponding total size of these samples. Note that the results for the pair of RIS and SUBSIM are almost the same as the latter just accelerates the sampling procedure and retains the sample outcomes. We can see from Figs. 9–12 that our 2-hop+ method generate substantially less samples than other baselines, despite that a 2-hop sample can have a larger expected size than an SKIS or RIS sample. This is because our 2-hop+ method can reduce the number of samples by avoiding generating samples with only 1-hop live paths which consist of a large portion of the samples. It can also be seen that the reduction in samples by our 2-hop+ method is more significant for the UNI model (Figs. 10 and 14) than other models because the UNI model generally has lower propagation probabilities and more samples with only 1-hop live paths are avoided generation in the sampling process. As a result, the total sample size returned by our 2-hop+ method is significantly reduced compared with SKIS and RIS, which makes our 2-hop+ method run faster than SKIS and RIS based methods. This confirms the results in Fig. 6. We can also see from Figs. 9–12 that different from the reverse samples consisting of only a few active nodes, each Monte Carlo simulation can activate a large number of nodes and the total size of the samples can be significantly larger than other reverse samples.



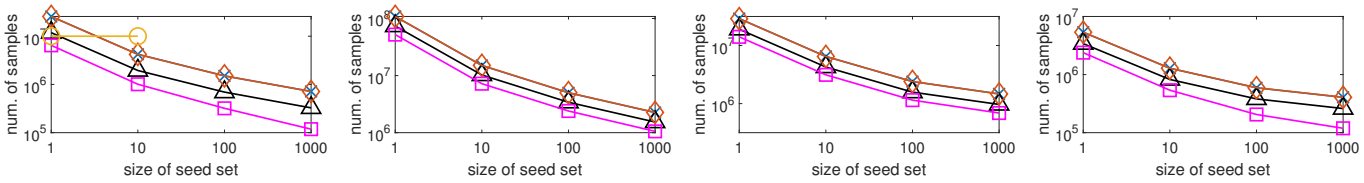
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 9: Number of samples generated for WC model.



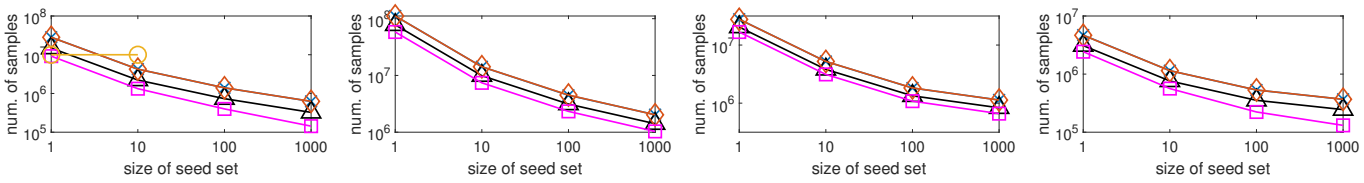
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 10: Number of samples generated for UNI model.



(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 11: Number of samples generated for EXP model.



(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

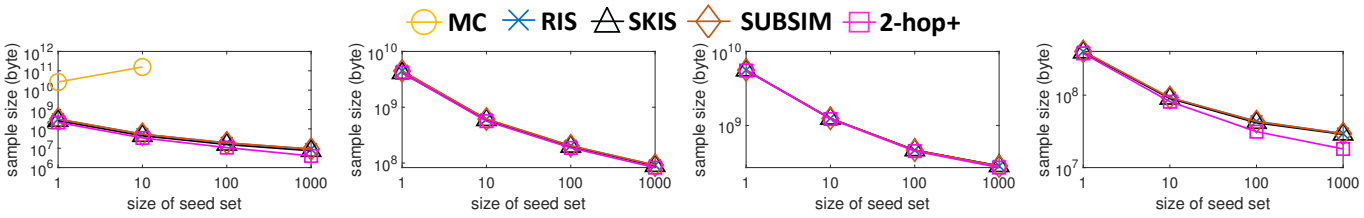
Fig. 12: Number of samples generated for WEI model.

TABLE 4: Average relative error (%) of influence estimation.

	Datasets	Google+				LiveJournal				Orkut				Twitter			
	k	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
WC	SKIS	2.3	1.9	2.1	2.0	2.4	2.4	2.2	2.1	2.3	2.5	2.2	2.0	2.4	2.2	2.0	1.5
	SUBSIM	2.3	2.2	2.3	2.1	2.4	2.3	2.5	2.2	2.4	2.4	2.2	2.1	2.4	2.3	2.2	1.8
	2-hop+	1.6	1.7	1.8	1.7	2.1	1.9	2.1	1.9	2.2	2.3	2.0	1.9	2.0	1.6	1.5	1.4
UNI	SKIS	2.1	2.1	2.2	2.1	2.1	2.2	2.1	2.0	2.5	2.5	2.1	2.2	2.2	2.3	1.9	1.9
	SUBSIM	2.5	2.3	2.5	2.7	2.4	2.5	2.5	2.4	2.6	2.7	2.1	2.6	2.2	2.3	2.5	2.1
	2-hop+	1.4	1.6	1.3	1.8	0.8	0.6	0.8	0.7	1.8	1.9	1.7	1.5	1.5	1.5	1.3	1.4
EXP	SKIS	2.0	2.2	2.0	1.7	2.3	2.2	2.5	2.3	2.7	2.3	2.1	2.0	2.1	2.1	1.9	1.8
	SUBSIM	2.3	2.3	2.7	2.1	2.7	2.4	2.4	2.5	2.7	2.5	2.3	2.3	2.2	2.1	2.0	2.0
	2-hop+	1.6	1.9	1.7	1.5	2.0	2.1	1.8	1.9	2.3	2.0	2.0	1.8	2.0	1.9	1.6	1.4
WEI	SKIS	2.4	2.3	2.1	1.9	2.3	2.5	2.4	2.1	2.2	2.2	2.0	2.3	2.2	2.0	1.8	1.6
	SUBSIM	2.8	2.4	2.4	2.2	2.4	2.7	2.6	2.3	2.2	2.3	2.1	2.3	2.2	2.2	2.0	1.8
	2-hop+	1.9	1.9	1.4	1.5	2.0	2.0	2.0	1.8	1.9	2.0	1.9	1.9	2.1	1.9	1.5	1.4

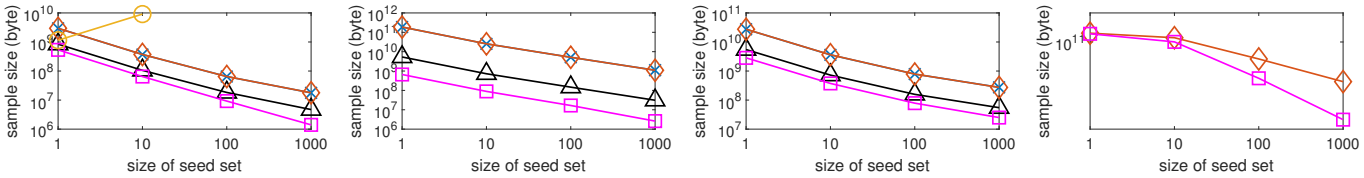
5.2.2 *Accuracy of Influence Estimation*

To compare the accuracy, we take the influence estimates generated under the (ε, δ) setting of $\varepsilon = 0.01$ and $\delta = 0.001$ as the ground-truth. We run SKIS, SUBSIM and our 2-hop+ methods each for 100 times under the (ε, δ) setting of $\varepsilon = 0.1$ and $\delta = 0.01$ and compare the influence estimates generated against the ground-truth. Note that the results of RIS and SUBSIM are almost the same, since SUBSIM just accelerates the sampling procedure of RIS and retains the sample outcomes of RIS. The results of RIS are omitted here. Table 4 shows the average relative error from the ground-truth for the three methods under the WC, UNI, EXP and WEI models respectively. As can be seen, our 2-hop+ method consistently achieves smaller relative errors compared with the other two baselines for all the datasets and seed set sizes tested. This is because our 2-hop+ method estimates a smaller portion of the influence spread via sampling than SKIS and SUBSIM. These results demonstrate that our 2-hop+ method can not only reduce the sample generation time but also improve the accuracy of influence estimation.



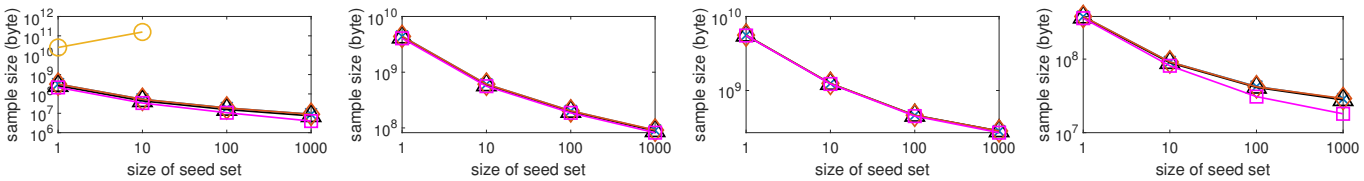
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 13: Total size of samples for WC model.



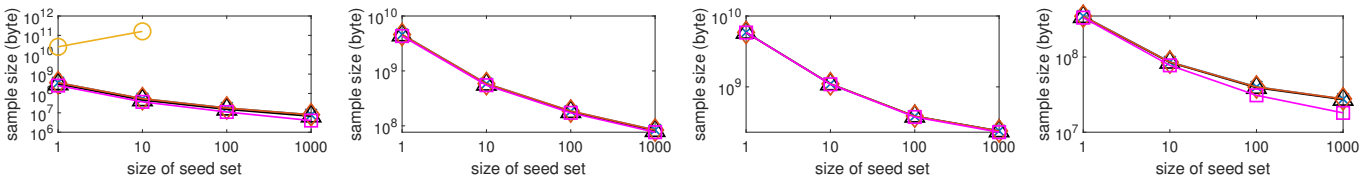
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 14: Total size of samples for UNI model.



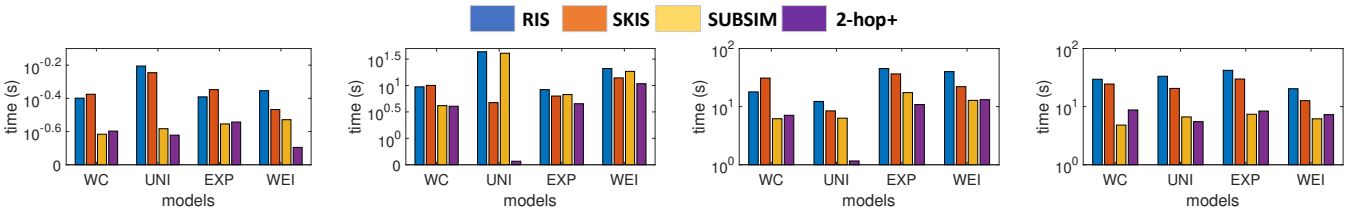
(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 15: Total size of samples for EXP model.



(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 16: Total size of samples for WEI model.



(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 17: Running time for influence maximization.

5.2.3 *Influence Maximization*

We apply the state-of-the-art technique in [34] to tackle the influence maximization problem where the number of samples is doubled in each round until the examined approximation guarantee is satisfied. The accuracy parameters of (ε, δ) are set as $(0.01, 0.001)$ and the target seed set size is set as 50. There are mainly two phases to tackle the influence maximization problem, i.e., the sampling phase and the seed selection phase. As our 2-hop+ sampling method generally requires less samples to achieve the accuracy guarantee compared with other methods, it takes relatively less time to build the samples under various models, where the sampling phase would cost a considerable portion of total time. To further reduce the overhead in the seed selection phase of our 2-hop+ method, we enlarge the number of samples in the initial round to reduce the number of iterations to reach the accuracy guarantee. As can be seen in Fig. 17, our 2-hop+ method can outperform RIS and SKIS in all cases and is comparable to SUBSIM. Similar to the performance in influence estimation, our 2-hop+ method can be much more efficient under the UNI model as 2-hop+ can significantly reduce the number of required samples to reach the accuracy guarantee. This shows that our 2-hop sampling method can work on the application of influence maximization efficiently.

5.2.4 *Initialization*

In our 2-hop+ sampling method, we need to precompute the parameters η_i^v, ϕ_i^v, $\Pr[T_i]$ $(1 \leq i \leq \ell_v)$ and β_v for each node $v \in V$ as discussed in Section 3. Similarly, the SKIS method [28] needs to precompute its parameters γ_v. Meanwhile, we use the alias method [42] to select the source node according to the probability distribution of $\Pr[\text{src} = v]$ which incurs some

TABLE 5: Time for initialization (seconds) under general cases.

Method	Google+	LiveJournal	Orkut	Twitter
SKIS	0.3	4.0	7.3	84.7
SUBSIM	0.3	1.0	4.1	30.4
2-hop+	0.9	12.9	26.7	300.3

additional preprocessing time for both methods. SUBSIM also needs to sort the probabilities of the incoming edges when they are non-uniform. Table 5 shows the running time of initialization under general cases. As expected, our 2-hop+ method has longer time of initialization than others since it needs to precompute more parameters but the increase in time is rather mild. Since the initialization of the parameters is one-off before generating samples to evaluate any number of seed sets, its impact on the running time of influence estimation is insignificant.

The additional experimental evaluations are presented in Appendix B of the supplementary file.

6 Related work

Kempe et al. [20] first proposed two well-known influence diffusion models, i.e., independent cascade (IC) and linear threshold (LT), based on which, they defined an influence maximization problem and developed an approximation solution using Monte Carlo simulations. Since then, there has been a large body of research on influence estimation or maximization [1, 4, 6, 8, 15, 21, 25, 33, 34, 36, 39, 40]. Chen et al. [7, 8] showed that computing the exact influence spread of a seed set is $\#$P-hard under both the IC and LT models. Borgs et al. [4] proposed the reverse influence sampling (RIS) method, which substantially improves the efficiency of influence estimation over Monte Carlo simulations. Many follow-up studies have made use of the RIS method to design efficient influence maximization algorithms with worst-case guarantees [27, 34, 39, 40]. In addition, the RIS method is extensively used to address a plethora of influence based optimization problems, including regret minimization [2], revenue maximization [3], profit maximization [32, 35, 37], adaptive influence maximization [17, 18], adaptive seed minimization [38], and adaptive profit maximization [19]. Nguyen et al. [28] improved the RIS method for influence estimation by proposing a SKIS method that generates only non-singular samples in which the RR set contains at least one additional node than the source node. In this way, the samples can have smaller variance and better concentration bounds. Meanwhile, some studies have observed that the majority of the influence spread is produced in the first few hops of propagation [1, 14, 36]. Our 2-hop sampling method takes advantage of this observation to enhance RIS by analytically computing the portion of influence spread within 1 hop of propagation and cutting the portion of influence spread to estimate experimentally through sampling. To further reduce the number of samples required, we develop some novel concentration bounds that improve the stopping rule algorithm to estimate the mean of random variables with (ε, δ)-approximation. Moreover, we also develop a SkipEdge algorithm to improve the sampling efficiency of our 2-hop RR sets. Our proposed techniques can be integrated with existing RIS based algorithms for the aforementioned optimization problems [2, 3, 4, 17, 18, 19, 27, 32, 34, 35, 37, 38, 39, 40] to improve their efficiency.

7 Conclusion

In this paper, we boost the sampling process for influence estimation in online social networks by two key techniques. First, we propose a 2-hop+ sampling method to generate 2-hop samples leading to tighter concentration bounds for imposing less samples and a SkipEdge algorithm to improve the sampling efficiency. Second, we improve the stopping rule algorithm with tighter threshold and thus further reduce the number of samples required. We show that our 2-hop+ method can significantly improve the efficiency of influence estimation by 1–2 orders of magnitudes compared to the state-of-the-art methods.

While this paper has focused on the IC model, our ideas and techniques can be easily extended and applied to the LT model as well. Under the LT model, the sampling process of the incoming edges to a given node can be accelerated by using the alias method [42] directly. To conduct 2-hop sampling, the probability distributions required can be derived similarly for the LT model. Finally, our stopping rule algorithm can be applied to any bounded random variables, including those representing the 2-hop samples generated under the LT model.

Acknowledgments

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 1 under Grant 2019-T1-002-042, by Hong Kong RGC ECS (No. 24203419), RGC CRF (No. C4158-20G), CUHK Direct Grant (No. 4055114), NSFC (No. U1936205), and by National Natural Science Foundation of China under Grant No. 72071139.

References

[1] A. Arora, S. Galhotra, and S. Ranu. Debunking the myths of influence maximization: An in-depth benchmarking study. In *Proc. ACM SIGMOD*, pages 651–666, 2017.

[2] C. Aslay, W. Lu, F. Bonchi, A. Goyal, and L. V. Lakshmanan. Viral marketing meets social advertising: Ad allocation with minimum regret. *Proc. VLDB Endowment*, 8(7):814–825, 2015.

[3] C. Aslay, F. Bonchi, L. V. Lakshmanan, and W. Lu. Revenue maximization in incentivized social advertising. *Proc. VLDB Endowment*, 10(11): 1238–1249, 2017.

[4] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proc. SODA*, pages 946–957, 2014.

[5] W. Chen and S.-H. Teng. Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality. In *Proc. WWW*, pages 967–976, 2017.

[6] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proc. ACM KDD*, pages 199–208, 2009.

[7] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. ACM KDD*, pages 1029–1038, 2010.

[8] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proc. IEEE ICDM*, pages 88–97, 2010.

[9] F. Chung and L. Lu. Concentration inequalities and martingale inequalities: A survey. *Internet Mathematics*, 3(1):79–127, 2006.

[10] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proc. ACM CIKM*, pages 629–638, 2014.

[11] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation. *SIAM Journal on Computing*, 29(5):1484–1496, 2000.

[12] T. N. Dinh and M. T. Thai. Assessing attack vulnerability in networks with uncertainty. In *Proc. IEEE INFOCOM*, pages 2380–2388, 2015.

[13] N. Du, L. Song, M. G. Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *Proc. NIPS*, pages 3147–3155, 2013.

[14] S. Goel, D. J. Watts, and D. G. Goldstein. The structure of online diffusion networks. In *Proc. ACM EC*, pages 623–638, 2012.

[15] A. Goyal, W. Lu, and L. V. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Proc. IEEE ICDM*, pages 211–220, 2011.

[16] Q. Guo, S. Wang, Z. Wei, and M. Chen. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proc. ACM SIGMOD*, pages 2167–2181, 2020.

[17] K. Han, K. Huang, X. Xiao, J. Tang, A. Sun, and X. Tang. Efficient algorithms for adaptive influence maximization. *Proc. VLDB Endowment*, 11(9):1029–1040, 2018.

[18] K. Huang, J. Tang, K. Han, X. Xiao, W. Chen, A. Sun, X. Tang, and A. Lim. Efficient approximation algorithms for adaptive influence maximization. *The VLDB Journal*, 29(6):1385–1406, 2020.

[19] K. Huang, J. Tang, X. Xiao, A. Sun, and A. Lim. Efficient approximation algorithms for adaptive target profit maximization. In *Proc. IEEE ICDE*, pages 649–660, 2020.

[20] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proc. ACM KDD*, pages 137–146, 2003.

[21] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *Proc. ICALP*, pages 1127–1138, 2005.

[22] D. E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[23] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proc. WWW*, pages 591–600, 2010.

[24] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology*, 8(1):1, 2016.

[25] B. Lucier, J. Oren, and Y. Singer. Influence at scale: Distributed computation of complex contagion in networks. In *Proc. ACM KDD*, pages 735–744, 2015.

[26] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, 1978.

[27] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proc. ACM SIGMOD*, pages 695–710, 2016.

[28] H. T. Nguyen, T. P. Nguyen, N. Phan, and T. N. Dinh. Importance sketching of influence dynamics in billion-scale networks. In *Proc. IEEE ICDM*, pages 337–346, 2017.

[29] H. T. Nguyen, T. P. Nguyen, T. N. Vu, and T. N. Dinh. Outward influence and cascade size estimation in billion-scale networks. *Proc. ACM on Measurement and Analysis of Computing Systems*, 1(1):20, 2017.

[30] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-i. Kawarabayashi. Dynamic influence analysis in evolving networks. *Proc. VLDB Endowment*, 9(12): 1077–1088, 2016.

[31] G. Sadeh, E. Cohen, and H. Kaplan. Sample complexity bounds for influence maximization. In *ITCS*, pages 29:1–29:36, 2020.

[32] J. Tang, X. Tang, and J. Yuan. Profit maximization for viral marketing in online social networks. In *Proc. IEEE ICNP*, pages 1–10, 2016.

[33] J. Tang, X. Tang, and J. Yuan. Influence maximization meets efficiency and effectiveness: A hop-based approach. In *Proc. IEEE/ACM ASONAM*, pages 64–71, 2017.

[34] J. Tang, X. Tang, X. Xiao, and J. Yuan. Online processing algorithms for influence maximization. In *Proc. ACM SIGMOD*, pages 991–1005, 2018.

[35] J. Tang, X. Tang, and J. Yuan. Towards profit maximization for online social network providers. In *Proc. IEEE INFOCOM*, pages 1178–1186, 2018.

[36] J. Tang, X. Tang, and J. Yuan. An efficient and effective hop-based approach for influence maximization in social networks. *Social Network Analysis and Mining*, 8(1):10, 2018.

[37] J. Tang, X. Tang, and J. Yuan. Profit maximization for viral marketing in online social networks: Algorithms and analysis. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1095–1108, 2018.

[38] J. Tang, K. Huang, X. Xiao, L. V. Lakshmanan, X. Tang, A. Sun, and A. Lim. Efficient approximation algorithms for adaptive seed minimization. In *Proc. ACM SIGMOD*, pages 1096–1113, 2019.

[39] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. ACM SIGMOD*, pages 75–86, 2014.

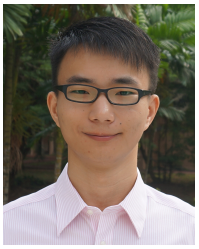
[40] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proc. ACM SIGMOD*, pages 1539–1554, 2015.

[41] A. Wald. *Sequential Analysis*. Wiley, 1947.

[42] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256, 1977.

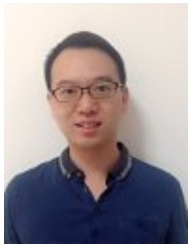
[43] Y. Zhu, J. Tang, and X. Tang. Pricing influential nodes in online social networks. *Proc. VLDB Endowment*, 13(10):1614–1627, 2020.

Yuqing Zhu received the BSc degree in computer science from Nanjing University of Science and Technology in 2017. He is currently a PhD student in Interdisciplinary Graduate School at Nanyang Technological University, Singapore.



Jing Tang (M'16) is an Assistant Professor with the Data Science and Analytics Thrust, The Hong Kong University of Science and Technology (HKUST). He received his Ph.D. degree from the Nanyang Technological University (NTU) in 2018 and his B.Eng. degree from the University of Science and Technology of China (USTC) in 2012. Prior to joining HKUST, he was a Research Assistant Professor at National University of Singapore (NUS) and a Research Fellow at Nanyang Technological University (NTU). His research interests include big data management and analytics, online social networks, distributed systems, and network economics. He received the Best Paper Award from the IEEE ICNP in 2014 and the Best-in-Session Presentation Award from the IEEE INFOCOM in 2018.

Xueyan Tang (M'04–SM'09) received the B.Eng. degree in computer science and engineering from Shanghai Jiao Tong University in 1998, and the Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 2003. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include distributed systems, cloud computing, mobile and pervasive computing. He has served as an associate editor of IEEE Transactions on Parallel and Distributed Systems, and a program co-chair of IEEE ICPADS 2012, CloudCom 2014 and ICDCS 2020. He is now serving as an associate editor of IEEE Transactions on Cloud Computing.



Sibo Wang joined the Chinese University of Hong Kong (CUHK) as an Assistant Professor in Dec 2018. He received the B.E. in Software Engineering in 2011 from Fudan University and PhD in Computer Science in 2016 from Nanyang Technological University, Singapore. Before joining CUHK, he spent one and half a year at the University of Queensland, Australia as a research fellow.

Andrew Lim is a researcher, educator, and deep tech specialist. His body of work draws from knowledge in computing, engineering, commerce and management to drive quantifiable performance improvements and to design and develop innovative and effective industrial systems and processes leading to digital transformation. In the past two decades or so, he has held tenured full professorships and headships in a number of top Universities in Asia and has won many international awards in research and applications. He was awarded the Thousand Talents Professorship in 2013 and the prestigious Returning Singapore Scientist Award in 2016.

APPENDIX A
MISSING PROOFS

Proof of Lemma 2. In 2-hop+ sampling, the source node is chosen from V according to the probability distribution of $\Pr[\mathrm{src} = v]$, which can be finished in $O(1)$ time using the alias method [42]. Let $\mathrm{EPT}_{\text{2-hop+}}$ denote the expected time complexity of 2-hop+ sampling for generating a 2-hop RR set $R_{\text{2-hop}}$, and $E_v(R_{\text{2-hop}})$ denote the number of live edges to v in $R_{\text{2-hop}}$. Then,

$$\mathrm{EPT}_{\text{2-hop+}} = O\Big(\mathbb{E}\Big[\sum\nolimits_{v\in R_{\text{2-hop}}}\Big(\log(\ell_v)\cdot\big(1+E_v(R_{\text{2-hop}})\big)\Big)\Big]\Big),$$

where the expectation is over the randomness of $R_{\text{2-hop}}$. Observe that given any node v, the expected number of active edges to v in all 2-hop RR sets containing v is μ_v, i.e., $\mathbb{E}[E_v(R_{\text{2-hop}}) \mid v \in R_{\text{2-hop}}] = \mu_v$. For notational convenience, we define $E_v(R_{\text{2-hop}}) = -1$ if $v \notin R_{\text{2-hop}}$. Thus,

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{v\in R_{\text{2-hop}}}\Big(\log(\ell_v)\cdot\big(1+E_v(R_{\text{2-hop}})\big)\Big)\Big]\\
&=\mathbb{E}\Big[\sum_{v\in V}\Big(\log(\ell_v)\cdot\big(1+E_v(R_{\text{2-hop}})\big)\Big)\Big]\\
&=\sum_{v\in V}\mathbb{E}\Big[\log(\ell_v)\cdot\big(1+E_v(R_{\text{2-hop}})\big)\Big]\\
&=\sum_{v\in V}\Big(\Pr[v\in R_{\text{2-hop}}]\cdot\log(\ell_v)\\
&\qquad\cdot\mathbb{E}\Big[\big(1+E_v(R_{\text{2-hop}})\big)\mid v\in R_{\text{2-hop}}\Big]\Big)\\
&=\sum_{v\in V}\Big(\Pr[v\in R_{\text{2-hop}}]\cdot\log(\ell_v)\cdot(1+\mu_v)\Big)\\
&=\frac{\mathrm{TP}}{\kappa}\cdot\sum_{v\in V}\Big(\sigma_2(\{v\})\cdot\frac{\log(\ell_v)\cdot(1+\mu_v)}{\mathrm{TP}}\Big)\\
&=\frac{\mathrm{TP}}{\kappa}\cdot\mathbb{E}[\sigma_2(\{v^\diamond\})].
\end{aligned}$$

This completes the proof. □

Proof of Lemma 4. With respect to $X_1, X_2, \ldots, X_{\lceil\theta\rceil}$, define a sequence of random variables $Y_1, Y_2, \ldots, Y_{\lfloor\theta\rfloor+1}$ with $Y_k = X_k$ for each $1 \le k \le \lfloor\theta\rfloor$, and $Y_{\lfloor\theta\rfloor+1} = (\theta - \lfloor\theta\rfloor)\cdot X_{\lceil\theta\rceil}$. Note that when θ is an integer, i.e., $\theta = \lfloor\theta\rfloor$, we have $Y_{\lfloor\theta\rfloor+1} = 0$. Thus, for each $1 \le k \le \lfloor\theta\rfloor$,

$$\mathbb{E}[Y_k \mid Y_1, Y_2, \ldots, Y_{k-1}] = \mathbb{E}[X_k \mid X_1, X_2, \ldots, X_{k-1}] = \mu_X.$$

Meanwhile,

$$\mathbb{E}[Y_{\lfloor\theta\rfloor+1} \mid Y_1, Y_2, \ldots, Y_{\lfloor\theta\rfloor}] = (\theta - \lfloor\theta\rfloor)\mu_X.$$

Define $M_t = \sum_{k=1}^{t}(Y_k - c_k\mu_X)$ for each $1 \le t \le \lfloor\theta\rfloor + 1$, where $c_k = 1$ if $1 \le k \le \lfloor\theta\rfloor$ and $c_{\lfloor\theta\rfloor+1} = \theta - \lfloor\theta\rfloor$. Thus, we can get that $\mathbb{E}[M_k \mid M_1, M_2, \ldots, M_{k-1}] = M_{k-1}$ and $\mathbb{E}[|M_k|] < \infty$ for any $k \le \lfloor\theta\rfloor + 1$. Therefore, according to Definition 1, $M_1, M_2, \ldots, M_{\lfloor\theta\rfloor+1}$ form a martingale. Similarly, $-M_1, -M_2, \ldots, -M_{\lfloor\theta\rfloor+1}$ also form a martingale.

In the martingale $M_1, M_2, \ldots, M_{\lfloor\theta\rfloor+1}$, we have $M_1 \le 1$ and $M_k - M_{k-1} \le 1$ for any $2 \le k \le \lfloor\theta\rfloor + 1$. In addition,

$$\begin{aligned}
&\mathrm{Var}[M_1] + \sum\nolimits_{k=2}^{\lfloor\theta\rfloor+1}\mathrm{Var}[M_k \mid M_1, M_2, \ldots, M_{k-1}]\\
&= \mathrm{Var}[Y_1] + \sum\nolimits_{k=2}^{\lfloor\theta\rfloor+1}\mathrm{Var}[Y_k \mid Y_1, Y_2, \ldots, Y_{k-1}]\\
&= \mathrm{Var}[X_1] + \sum\nolimits_{k=2}^{\lfloor\theta\rfloor}\mathrm{Var}[X_k \mid X_1, X_2, \ldots, X_{k-1}]\\
&\quad+(\theta-\lfloor\theta\rfloor)^2\cdot\mathrm{Var}[X_{\lceil\theta\rceil}\mid X_1, X_2\ldots, X_{\lfloor\theta\rfloor}]\\
&\le \lfloor\theta\rfloor\mu_X(1-\mu_X) + (\theta-\lfloor\theta\rfloor)^2\mu_X(1-\mu_X)\\
&\le \lfloor\theta\rfloor\mu_X(1-\mu_X) + (\theta-\lfloor\theta\rfloor)\mu_X(1-\mu_X)\\
&= \theta\mu_X(1-\mu_X).
\end{aligned}$$

Since $\mathbb{E}[M_{\lfloor\theta\rfloor+1}] = 0$, by Lemma 3, we have

$$\begin{aligned}
\Pr[\bar{X} \ge \mu_X + \lambda] &= \Pr\big[\sum\nolimits_{k=1}^{\lfloor\theta\rfloor+1} Y_k - \theta\cdot\mu_X \ge \theta\lambda\big]\\
&= \Pr[M_{\lfloor\theta\rfloor+1} \ge \theta\lambda]\\
&\le \exp\Big(-\frac{\theta^2\lambda^2}{\frac{2}{3}\theta\lambda + 2\theta\mu_X(1-\mu_X)}\Big)\\
&\le \exp\Big(-\frac{\lambda^2\theta}{2\mu_X + \frac{2}{3}\lambda}\Big).
\end{aligned}$$

In the martingale $-M_1, -M_2, \ldots, -M_{\lfloor\theta\rfloor+1}$, we have $-M_1 \le \mu_X$ and $-M_k + M_{k-1} \le \mu_X$ for any $2 \le k \le \lfloor\theta\rfloor + 1$. In addition,

$$\begin{aligned}
&\mathrm{Var}[-M_1] + \sum\nolimits_{k=2}^{\lfloor\theta\rfloor+1}\mathrm{Var}[-M_k \mid -M_1, -M_2, \ldots, -M_{k-1}]\\
&= \mathrm{Var}[Y_1] + \sum\nolimits_{k=2}^{\lfloor\theta\rfloor+1}\mathrm{Var}[Y_k \mid Y_1, Y_2, \ldots, Y_{k-1}]\\
&\le \theta\mu_X(1-\mu_X).
\end{aligned}$$

If $\lambda > \mu_X$, (13) is trivial. If $\lambda \le \mu_X$, since $\mathbb{E}[-M_{\lfloor\theta\rfloor+1}] = 0$, by Lemma 3, we have

$$\begin{aligned}
\Pr[\bar{X} \le \mu_X - \lambda] &= \Pr\big[\sum\nolimits_{k=1}^{\lfloor\theta\rfloor+1} Y_k - \theta\cdot\mu_X \le -\theta\lambda\big]\\
&= \Pr[-M_{\lfloor\theta\rfloor+1} \ge \theta\lambda]\\
&\le \exp\Big(-\frac{\theta^2\lambda^2}{\frac{2}{3}\mu_X\theta\lambda + 2\theta\mu_X(1-\mu_X)}\Big)\\
&\le \exp\Big(-\frac{\lambda^2\theta}{2\mu_X + \frac{2}{3}\mu_X^2 - 2\mu_X^2}\Big)\\
&\le \exp\Big(-\frac{\lambda^2\theta}{2\mu_X}\Big).
\end{aligned}$$

This completes the proof. □

Proof of Theorem 3. To prove (14), we prove the following two inequalities:

$$\Pr[\tilde{\mu}_Z < (1-\varepsilon)\mu_Z] \le \frac{\delta}{2}, \tag{16}$$

$$\Pr[\tilde{\mu}_Z > (1+\varepsilon)\mu_Z] \le \frac{\delta}{2}. \tag{17}$$

When Algorithm 4 terminates, we have

$$\tilde{\mu}_Z = \frac{\Upsilon}{\theta}, \quad \text{and} \quad \sum\nolimits_{k=1}^{\lfloor\theta\rfloor} Z_k + (\theta - \lfloor\theta\rfloor)Z_{\lceil\theta\rceil} = \Upsilon.$$

Let $Z'_k = \frac{Z_k - a}{b-a}$ so that $Z'_k \in [0,1]$. Define $\mu_{Z'} = \mathbb{E}[Z'_k] = \frac{\mu_Z - a}{b-a}$. Let $\vartheta_1 = \frac{\Upsilon}{(1-\varepsilon)\mu_Z}$. Then, we have

$$\begin{aligned}
&\Pr[\tilde{\mu}_Z < (1-\varepsilon)\mu_Z]\\
&= \Pr[\Upsilon < (1-\varepsilon)\mu_Z\theta]\\
&= \Pr[\vartheta_1 < \theta]\\
&\le \Pr\Big[\sum_{k=1}^{\lfloor\vartheta_1\rfloor} Z_k + (\vartheta_1 - \lfloor\vartheta_1\rfloor)Z_{\lceil\vartheta_1\rceil}\\
&\qquad\le \sum_{k=1}^{\lfloor\theta\rfloor} Z_k + (\theta - \lfloor\theta\rfloor)Z_{\lceil\theta\rceil}\Big]
\end{aligned}$$

$$= \Pr\Big[\sum\nolimits_{k=1}^{\lfloor\vartheta_1\rfloor} Z_k + (\vartheta_1 - \lfloor\vartheta_1\rfloor)Z_{\lceil\vartheta_1\rceil} \le \Upsilon\Big]$$
$$= \Pr\Big[\frac{\sum_{k=1}^{\lfloor\vartheta_1\rfloor} Z_k + (\vartheta_1 - \lfloor\vartheta_1\rfloor)Z_{\lceil\vartheta_1\rceil}}{\vartheta_1} \le \frac{\Upsilon}{\vartheta_1}\Big]$$
$$= \Pr\Big[\frac{\sum_{k=1}^{\lfloor\vartheta_1\rfloor} Z_k + (\vartheta_1 - \lfloor\vartheta_1\rfloor)Z_{\lceil\vartheta_1\rceil}}{\vartheta_1} \le (1-\varepsilon)\mu_Z\Big]$$
$$= \Pr\Big[\frac{\sum_{k=1}^{\lfloor\vartheta_1\rfloor} Z'_k + (\vartheta_1 - \lfloor\vartheta_1\rfloor)Z'_{\lceil\vartheta_1\rceil}}{\vartheta_1} \le \mu_{Z'} - \varepsilon(\mu_{Z'} + \frac{a}{b-a})\Big].$$

Next, we prove (17). Let $\vartheta_2 = \frac{\Upsilon}{(1+\varepsilon)\mu_Z}$. Similarly, we have

$$\Pr[\tilde{\mu}_Z > (1+\varepsilon)\mu_Z]$$
$$= \Pr[\Upsilon > (1+\varepsilon)\mu_Z\theta]$$
$$= \Pr[\vartheta_2 > \theta]$$
$$\le \Pr\Big[\sum\nolimits_{k=1}^{\lfloor\vartheta_2\rfloor} Z_k + (\vartheta_2 - \lfloor\vartheta_2\rfloor)Z_{\lceil\vartheta_2\rceil} \ge \sum\nolimits_{k=1}^{\lfloor\theta\rfloor} Z_k + (\theta - \lfloor\theta\rfloor)Z_{\lceil\vartheta\rceil}\Big]$$
$$= \Pr\Big[\sum\nolimits_{k=1}^{\lfloor\vartheta_2\rfloor} Z_k + (\vartheta_2 - \lfloor\vartheta_2\rfloor)Z_{\lceil\vartheta_2\rceil} \ge \Upsilon\Big]$$
$$= \Pr\Big[\frac{\sum_{k=1}^{\lfloor\vartheta_2\rfloor} Z_k + (\vartheta_2 - \lfloor\vartheta_2\rfloor)Z_{\lceil\vartheta_2\rceil}}{\vartheta_2} \ge \frac{\Upsilon}{\vartheta_2}\Big]$$
$$= \Pr\Big[\frac{\sum_{k=1}^{\lfloor\vartheta_2\rfloor} Z_k + (\vartheta_2 - \lfloor\vartheta_2\rfloor)Z_{\lceil\vartheta_2\rceil}}{\vartheta_2} \ge (1+\varepsilon)\mu_Z\Big]$$
$$= \Pr\Big[\frac{\sum_{k=1}^{\lfloor\vartheta_2\rfloor} Z'_k + (\vartheta_2 - \lfloor\vartheta_2\rfloor)Z'_{\lceil\vartheta_2\rceil}}{\vartheta_2} \ge \mu_{Z'} + \varepsilon(\mu_{Z'} + \frac{a}{b-a})\Big].$$

Applying (12) of Lemma 4, we obtain that

$$\Pr[\tilde{\mu}_Z > (1+\varepsilon)\mu_Z]$$
$$\le \exp\Big(-\frac{\varepsilon^2(\mu_{Z'} + \frac{a}{b-a})^2\vartheta_2}{2\mu_{Z'} + \frac{2}{3}\varepsilon(\mu_{Z'} + \frac{a}{b-a})}\Big)$$
$$= \exp\Big(-\frac{\varepsilon^2\mu_Z\vartheta_2}{2(b-a)(\frac{\mu_{Z'}}{\mu_{Z'}+\frac{a}{b-a}} + \frac{1}{3}\varepsilon)}\Big)$$
$$\le \exp\Big(-\frac{\varepsilon^2\mu_Z\vartheta_2}{2(b-a)(\frac{b-a}{b} + \frac{1}{3}\varepsilon)}\Big)$$
$$= \exp\Big(-\frac{\varepsilon^2\Upsilon}{2(b-a)(\frac{b-a}{b} + \frac{1}{3}\varepsilon)(1+\varepsilon)}\Big)$$
$$= \frac{\delta}{2}.$$

Applying (13) of Lemma 4, we obtain that

$$\Pr[\tilde{\mu}_Z \le (1-\varepsilon)\mu_Z]$$
$$\le \exp\Big(-\frac{\varepsilon^2(\mu_{Z'} + \frac{a}{b-a})^2\vartheta_1}{2\mu_{Z'}}\Big)$$
$$= \exp\Big(-\frac{\varepsilon^2(1 + \frac{a}{(b-a)\mu_{Z'}})\mu_Z\vartheta_1}{2(b-a)}\Big)$$
$$\le \exp\Big(-\frac{\varepsilon^2\frac{b}{b-a}\mu_Z\vartheta_1}{2(b-a)}\Big)$$
$$= \exp\Big(-\frac{\varepsilon^2\frac{b}{b-a}\Upsilon}{2(b-a)(1-\varepsilon)}\Big)$$
$$= \exp\Big(-\frac{(1+\varepsilon)(1+\frac{b\varepsilon}{3(b-a)})\ln(\frac{2}{\delta})}{(1-\varepsilon)}\Big)$$
$$\le \frac{\delta}{2}.$$

By the union bound, (16) and (17) give rise to (14). Meanwhile, based on (16), we have

$$\Pr\Big[\theta > \frac{\Upsilon}{(1-\varepsilon)\mu_Z}\Big] = \Pr[\tilde{\mu}_Z < (1-\varepsilon)\mu_Z] \le \frac{\delta}{2},$$

which completes the proof of (15). □

Proof of Theorem 2. For a random RR set R_{RIS} generated by the RIS method, let $\mathcal{E}'_a$ and $\mathcal{E}'_b$ denote the following events

$$\mathcal{E}'_a\colon R_{\mathsf{RIS}} \cap S \neq \emptyset \wedge \mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset,$$
$$\mathcal{E}'_b\colon R_{\mathsf{RIS}} \cap S \neq \emptyset \wedge \mathrm{h}_{2^*}(R_{\mathsf{RIS}}) \neq \emptyset.$$

Therefore,

$$\frac{\sigma(S)}{|V|} = \Pr[S \cap R_{\mathsf{RIS}} \neq \emptyset] = \Pr[\mathcal{E}'_a] + \Pr[\mathcal{E}'_b]. \quad (18)$$

An RR set R_{RIS} is a 2-hop RR set if and only if $\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) \neq \emptyset$. Thus, when event $\mathcal{E}'_b$ occurs, R_{RIS} is a 2-hop RR set, indicating that

$$\Pr[\mathcal{E}'_b] = \Pr[R_{\text{2-hop}} \cap S \neq \emptyset] \cdot \frac{\kappa}{|V|}. \quad (19)$$

On the other hand, when event $\mathcal{E}'_a$ occurs, $\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset$ and the source node $v = \mathrm{src}(R_{\mathsf{RIS}})$ must satisfy either (i) $v \in S$ or (ii) $v \in N_S \setminus S$ and $\mathrm{h}_1(R_{\mathsf{RIS}}) \cap S \neq \emptyset$. In fact, we know that the probability of R_{RIS} generated from source node v satisfying $\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset$ is

$$\Pr[\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset \mid \mathrm{src}(R_{\mathsf{RIS}}) = v] = 1 - \beta_v.$$

As a result, given $v \in S$,

$$\Pr[\mathcal{E}'_a \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v]$$
$$= \Pr[\mathrm{src}(R_{\mathsf{RIS}}) = v] \cdot \Pr[\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset \mid \mathrm{src}(R_{\mathsf{RIS}}) = v]$$
$$= \frac{1-\beta_v}{|V|}.$$

Moreover, $\alpha_{u,v}$ is the probability that there is no 2-hop live path going through node u from source node v. Thus, conditional on $\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset$, for any given $u \in I_v$, the probability of edge (u,v) being blocked is

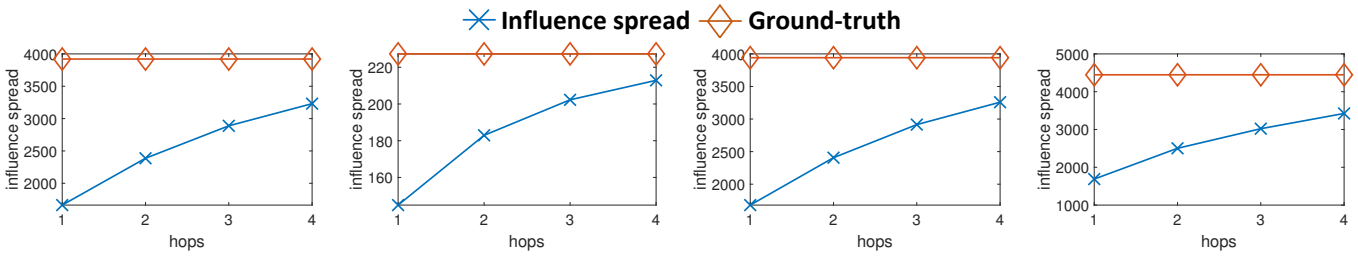
$$\Pr[u \in \mathrm{h}_1(R_{\mathsf{RIS}}) \mid \mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v] = \frac{1-p_{u,v}}{\alpha_{u,v}}.$$

Therefore, the probability that there exists a node $u \in I_v \cap S$ satisfying edge (u,v) being live is

$$\Pr[\mathrm{h}_1(R_{\mathsf{RIS}}) \cap S \neq \emptyset \mid \mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v]$$
$$= 1 - \prod_{u \in I_v \cap S} \frac{1-p_{u,v}}{\alpha_{u,v}}.$$

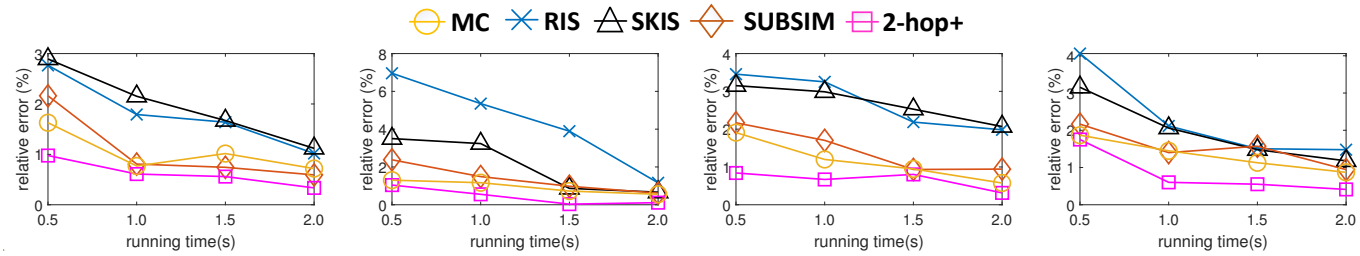
As a result, given $v \in N_S \setminus S$,

$$\Pr[\mathcal{E}'_a \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v]$$
$$= \Pr[\mathrm{src}(R_{\mathsf{RIS}}) = v] \cdot \Pr[\mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset \mid \mathrm{src}(R_{\mathsf{RIS}}) = v]$$
$$\cdot \Pr[\mathrm{h}_1(R_{\mathsf{RIS}}) \cap S \neq \emptyset \mid \mathrm{h}_{2^*}(R_{\mathsf{RIS}}) = \emptyset \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v]$$



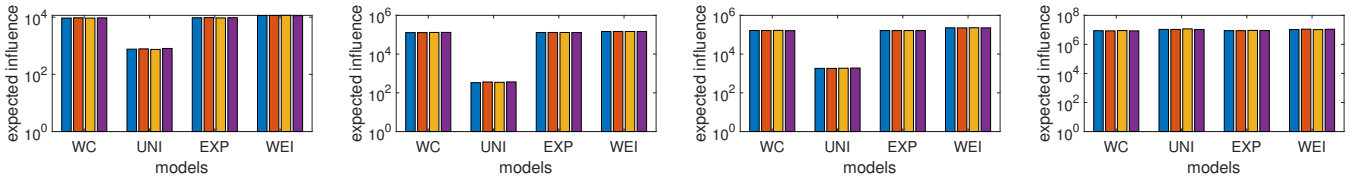
(a) Google+ (WC) (b) Google+ (UNI) (c) Google+ (EXP) (d) Google+ (WEI)

Fig. 18: Influence spread within hops.



(a) Google+ (WC) (b) Google+ (UNI) (c) Google+ (EXP) (d) Google+ (WEI)

Fig. 19: Empirical relative error.



(a) Google+ (b) LiveJournal (c) Orkut (d) Twitter

Fig. 20: Expected influence spread of influence maximization.

$$= \frac{1-\beta_v}{|V|} \cdot \Big(1 - \prod_{u \in I_v \cap S} \frac{1-p_{u,v}}{\alpha_{u,v}}\Big).$$

Finally, we have

$$\begin{aligned}
&\Pr[\mathcal{E}'_a] \\
&= \sum_{v \in S} \Pr[\mathcal{E}'_a \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v] \\
&\quad + \sum_{v \in N_S \setminus S} \Pr[\mathcal{E}'_a \wedge \mathrm{src}(R_{\mathsf{RIS}}) = v] \\
&= \sum_{v \in S} \frac{1-\beta_v}{|V|} + \sum_{v \in N_S \setminus S} \Big(\frac{1-\beta_v}{|V|} \cdot \Big(1 - \prod_{u \in I_v \cap S} \frac{1-p_{u,v}}{\alpha_{u,v}}\Big)\Big) \\
&= \frac{1}{|V|} \sum_{v \in N_S \cup S} \Big((1-\beta_v) \cdot \Big(1 - \prod_{u \in (I_v \cup \{v\}) \cap S} \frac{1-p_{u,v}}{\alpha_{u,v}}\Big)\Big).
\end{aligned} \tag{20}$$

Putting it all together of (18)–(20) completes the proof. □

APPENDIX B
ADDITIONAL EVALUATIONS

Influence Spread in Hops. Fig. 18 shows the influence spread of 10 top-degree nodes within hops of propagation on the dataset of Google+. The ground-truth influence spread is obtained by a (0.01,0.001)-estimate. As can be seen from Fig. 18, when the diffusion steps $\tau \leq 4$, the influence spread is significantly smaller than the ground truth. Applying the diffusion step constraint with a small value of τ can improve the efficiency but at the cost of accuracy, which is far from the desire of (ε, δ)-estimate of influence spread.

Empirical Relative Error. To measure the empirical quality of estimation, we run different sampling algorithms on the dataset of Google+ with 10 top-degree nodes as seeds and use a (0.01,0.001)-estimate as the ground-truth. We plot the curves of empirical relative error delivered by different sampling methods. As shown in Fig. 19, the empirical accuracy of Monte-Carlo simulation is comparable to other reverse sampling methods when running within a given time limit. However, it remains a challenge for devising practical MC-based algorithms with rigorous (ε, δ) approximation guarantees.

Influence Maximization. Fig. 20 shows that our 2-hop sampling method can achieve similar expected influence spread compared with other baselines for influence maximization.

Statistical Significance. We perform the statistical significance test for time, space complexity and accuracy results as follows. We run our algorithm together with the baselines for 100 times to obtain the $(0.01, 0.001)$-estimate on Google+ and record the relative standard deviation of the number of samples, the running time and the total size of samples in Table 6. The performance on other datasets is similar. It can be seen that the standard deviation is small among 100 runs which can hardly affect the relative performance among different algorithms and the result is stable and reliable.

We calculate the confidence interval characterizing the range of relative error where the influence estimation will lie in with probability 0.95. As can be seen from Tables 7–10, our 2-hop+ method can deliver an estimation of influence spread falling in an interval with smaller relative error compared with other methods.

TABLE 6: Standard deviation (%).

	Metrics	Num. of samples				Running time				Total size			
	k	1	10	100	1000	1	10	100	1000	1	10	100	1000
WC	RIS	0.18	0.15	0.15	0.13	3.5	4.2	4.9	1.5	0.18	0.15	0.12	0.13
	SUBSIM	0.14	0.14	0.15	0.14	2.5	1.5	3.9	1.4	0.14	0.12	0.12	0.12
	SKIS	0.15	0.14	0.13	0.12	2.9	5.8	3.3	5.4	0.15	0.15	0.13	0.15
	2-hop+	0.11	0.09	0.1	0.09	2.7	3.3	2.9	4.5	0.12	0.09	0.14	0.17
UNI	RIS	0.14	0.15	0.14	0.13	0.9	1.4	1.6	4.9	0.12	0.15	0.13	0.13
	SUBSIM	0.13	0.14	0.16	0.17	1.3	0.8	1.3	4.6	0.14	0.13	0.14	0.15
	SKIS	0.15	0.16	0.15	0.12	1.4	3.6	2.1	1.7	0.15	0.16	0.14	0.14
	2-hop+	0.08	0.09	0.09	0.08	2.8	3.2	1.5	1.8	0.08	0.08	0.12	0.26
EXP	RIS	0.14	0.15	0.14	0.16	0.4	1.4	6.0	4.9	0.14	0.15	0.12	0.13
	SUBSIM	0.16	0.15	0.15	0.12	2.3	0.9	1.7	3.2	0.16	0.15	0.12	0.11
	SKIS	0.16	0.13	0.12	0.13	2.5	4.5	4.3	4.2	0.15	0.13	0.12	0.14
	2-hop+	0.13	0.11	0.11	0.09	2.9	1.8	2.9	2.5	0.12	0.11	0.11	0.18
WEI	RIS	0.15	0.15	0.14	0.13	0.17	1.9	1.6	3.6	0.15	0.13	0.13	0.11
	SUBSIM	0.15	0.17	0.14	0.14	1.0	1.9	3.8	2.4	0.15	0.16	0.11	0.13
	SKIS	0.13	0.14	0.15	0.11	0.9	3.0	2.6	1.9	0.13	0.14	0.16	0.13
	2-hop+	0.14	0.11	0.09	0.10	3.5	1.9	1.6	2.2	0.14	0.11	0.10	0.14

TABLE 7: Confidence interval of relative error (%) for influence estimation on Google+.

Models	**WC**				**UNI**			
k	1	10	100	1000	1	10	100	1000
SUBSIM	[2.0,2.7]	[1.8,2.5]	[2.0,2.6]	[1.8,2.4]	[2.2,2.8]	[2.0,2.7]	[2.1,2.8]	[2.3,3.0]
SKIS	[2.0,2.6]	[1.6,2.3]	[1.8,2.4]	[1.7,2.3]	[1.8,2.4]	[1.8,2.4]	[1.9,2.5]	[1.8,2.4]
2-hop+	[1.4,1.8]	[1.4,1.9]	[1.6,2.1]	[1.5,2.0]	[1.2,1.6]	[1.4,1.8]	[1.1,1.4]	[1.6,2.0]
Models	**EXP**				**WEI**			
SUBSIM	[2.0,2.6]	[1.9,2.7]	[2.3,3.0]	[1.8,2.4]	[2.4,3.1]	[2.0,2.7]	[2.0,2.7]	[1.8,2.5]
SKIS	[1.7,2.4]	[1.8,2.5]	[1.7,2.3]	[1.4,1.9]	[2.1,2.7]	[1.9,2.5]	[1.7,2.4]	[1.6,2.2]
2-hop+	[1.4,1.8]	[1.6,2.2]	[1.5,1.9]	[1.3,1.6]	[1.7,2.1]	[1.6,2.1]	[1.2,1.6]	[1.3,1.7]

TABLE 8: Confidence interval of relative error (%) for influence estimation on LiveJournal.

Models	**WC**				**UNI**			
k	1	10	100	1000	1	10	100	1000
SUBSIM	[2.0,2.7]	[1.9,2.6]	[2.1,2.9]	[1.9,2.5]	[2.1,2.8]	[2.1,2.8]	[2.1,2.9]	[2.1,2.8]
SKIS	[2.1,2.7]	[2.0,2.7]	[1.9,2.6]	[1.8,2.4]	[1.8,2.4]	[1.9,2.5]	[1.8,2.4]	[1.7,2.3]
2-hop+	[1.7,2.4]	[1.6,2.2]	[1.8,2.4]	[1.7,2.2]	[0.7,0.9]	[0.5,0.7]	[0.7,0.9]	[0.6,0.8]
Models	**EXP**				**WEI**			
SUBSIM	[2.3,3.1]	[2.1,2.8]	[2.1,2.7]	[2.2,2.8]	[2.1,2.8]	[2.3,3.0]	[2.2,3.0]	[2.0,2.6]
SKIS	[2.0,2.6]	[1.9,2.5]	[2.1,2.8]	[1.9,2.6]	[2.0,2.7]	[2.1,2.8]	[2.1,2.7]	[1.8,2.5]
2-hop+	[1.7,2.3]	[1.8,2.3]	[1.5,2.1]	[1.6,2.1]	[1.7,2.3]	[1.8,2.3]	[1.7,2.2]	[1.5,2.0]

TABLE 9: Confidence interval of relative error (%) for influence estimation on Orkut.

Models	**WC**				**UNI**			
k	1	10	100	1000	1	10	100	1000
SUBSIM	[2.0,2.8]	[2.0,2.7]	[1.9,2.6]	[1.8,2.4]	[2.2,2.9]	[2.3,3.1]	[1.7,2.4]	[2.2,3.0]
SKIS	[2.0,2.6]	[2.1,2.8]	[1.9,2.6]	[1.7,2.3]	[2.1,2.9]	[2.1,2.8]	[1.7,2.4]	[1.9,2.4]
2-hop+	[1.9,2.5]	[2.0,2.7]	[1.6,2.3]	[1.6,2.2]	[1.6,2.0]	[1.6,2.1]	[1.4,1.9]	[1.3,1.7]
Models	**EXP**				**WEI**			
SUBSIM	[2.3,3.1]	[2.0,2.9]	[2.0,2.7]	[1.9,2.6]	[1.9,2.6]	[1.9,2.7]	[1.7,2.5]	[1.9,2.7]
SKIS	[2.3,3.0]	[1.9,2.7]	[1.7,2.4]	[1.6,2.3]	[1.8,2.6]	[1.8,2.5]	[1.7,2.4]	[1.9,2.6]
2-hop+	[2.0,2.7]	[1.7,2.3]	[1.7,2.3]	[1.6,2.1]	[1.6,2.3]	[1.6,2.3]	[1.6,2.2]	[1.7,2.1]

TABLE 10: Confidence interval of relative error (%) for influence estimation on Twitter.

Models	**WC**				**UNI**			
k	1	10	100	1000	1	10	100	1000
SUBSIM	[2.1,2.7]	[2.0,2.6]	[1.9,2.5]	[1.6,2.1]	[1.9,2.5]	[1.9,2.7]	[2.1,2.8]	[1.8,2.4]
SKIS	[2.1,2.7]	[1.9,2.6]	[1.7,2.3]	[1.3,1.7]	[1.9,2.5]	[1.9,2.6]	[1.6,2.1]	[1.6,2.1]
2-hop+	[1.8,2.3]	[1.4,1.9]	[1.2,1.7]	[1.2,1.6]	[1.3,1.8]	[1.3,1.7]	[1.2,1.5]	[1.2,1.6]
Models	**EXP**				**WEI**			
SUBSIM	[1.9,2.7]	[1.7,2.5]	[1.7,2.3]	[1.7,2.2]	[1.9,2.6]	[1.8,2.5]	[1.7,2.3]	[1.5,2.0]
SKIS	[1.7,2.5]	[1.8,2.4]	[1.6,2.1]	[1.6,2.0]	[1.9,2.5]	[1.7,2.3]	[1.5,2.0]	[1.4,1.8]
2-hop+	[1.7,2.3]	[1.6,2.1]	[1.4,1.9]	[1.2,1.6]	[1.8,2.4]	[1.6,2.2]	[1.3,1.7]	[1.2,1.6]