Influence Maximization Revisited: Efficient Sampling with Bound Tightened

QINTIAN GUO, The Chinese University of Hong Kong, China
SIBO WANG*, The Chinese University of Hong Kong, China
ZHEWEI WEI, Gaoling School of Artificial Intelligence, Renmin University of China, China
WENQING LIN, Tencent, China
JING TANG, The Hong Kong University of Science and Technology (Guangzhou), China and The Hong Kong University of Science and Technology, China

Given a social network G with n nodes and m edges, a positive integer k, and a cascade model C, the *influence maximization (IM)* problem asks for k nodes in G such that the expected number of nodes influenced by the k nodes under cascade model C is maximized. The state-of-the-art approximate solutions run in $O(k(n+m)\log n/\epsilon^2)$ expected time while returning a $(1-1/e-\epsilon)$ approximate solution with at least $1-1/n$ probability. A key phase of these IM algorithms is the random *reverse reachable (RR)* set generation, and this phase significantly affects the efficiency and scalability of the state-of-the-art IM algorithms.

In this paper, we present a study on this key phase and propose an efficient random RR set generation algorithm under IC model. With the new algorithm, we show that the expected running time of existing IM algorithms under IC model can be improved to $O(k \cdot n \log n/\epsilon^2)$, when for any node v, the total weight of its incoming edges is no larger than a constant. For general IC model where the weights are skewed, we present a sampling algorithm SKIP. To the best of our knowledge, it is the first index-free algorithm that achieves the optimal time complexity of the sorted subset sampling problem.

Moreover, existing approximate IM algorithms suffer from scalability issues in high influence networks where the size of random RR sets is usually quite large. We tackle this challenging issue by reducing the average size of random RR sets without sacrificing the approximation guarantee. The proposed solution is orders of magnitude faster than states of the art as shown in our experiment.

Besides, we investigate the issues of forward propagation and derive its time complexity with our proposed subset sampling techniques. We also present a heuristic condition to indicate when the forward propagation approach should be utilized to estimate the expected influence of a given seed set.

CCS Concepts: • **Mathematics of computing → Graph algorithms**.

*Sibo Wang and Zhewei Wei are the corresponding authors.

Sibo Wang is supported by Hong Kong RGC ECS (Grant No. 24203419), Hong Kong RGC CRF (Grant No. C4158-20G), Hong Kong ITC ITF (Grant No. MRP/071/20X), CUHK Direct Grant (Grant No. 4055181) and NSFC of China (Grant No. U1936205). Zhewei Wei's work was partially done at MOE Key Lab of Data Engineering and Knowledge Engineering, and Peng Cheng Laboratory, and was supported in part by National Natural Science Foundation of China (No. 61932001 and No. 61972401), Beijing Natural Science Foundation (No. 4222028), and the major key project of PCL (PCL2021A12). Jing Tang's work is partially supported by HKUST(GZ) under a Startup Grant.
Authors' addresses: Qintian Guo, qtguo@se.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, China; Sibo Wang, The Chinese University of Hong Kong, Hong Kong, China, swang@se.cuhk.edu.hk; Zhewei Wei, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China, zhewei@ruc.edu.cn; Wenqing Lin, Tencent, Shenzhen, China, edwlin@tencent.com; Jing Tang, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China and The Hong Kong University of Science and Technology, Hong Kong, China, jingtang@ust.hk.

0362-5915/2022/1-ART1 $15.00
https://doi.org/10.1145/3533817

Additional Key Words and Phrases: Influence Maximization; Sampling

ACM Reference Format:
Qintian Guo, Sibo Wang, Zhewei Wei, Wenqing Lin, and Jing Tang. 2022. Influence Maximization Revisited: Efficient Sampling with Bound Tightened. *ACM Trans. Datab. Syst.* 1, 1, Article 1 (January 2022), 45 pages. https://doi.org/10.1145/3533817

1 INTRODUCTION

In social networks, *cascade* models the word-of-mouth effect that users adopt certain products, take up some opinions or receive certain information due to the influence of their friends. Given a social network G with n nodes and m edges, a positive integer k, and a cascade model C, the *influence maximization (IM)* problem asks for k nodes in G that can infect the largest number of nodes in cascade model C. IM finds important applications in viral marketing, a marketing strategy that a company provides their product freely to a few influential users in social networks, in the hope that they will recommend the product to their friends.

Kempe et al. [30] present the first seminal work on IM, and show that finding k users which maximizes the influence is NP-hard. They consider two popular cascade models, the *Independent-Cascade (IC)* model and *Linear-Threshold (LT)* model, and provide a general greedy algorithm that provides $(1 - 1/e - \epsilon)$-approximate solutions for both cascade models. However, the proposed solution requires $\Omega(k \cdot m \cdot n \cdot poly(1/\epsilon))$ running time and is prohibitively expensive on large social networks. A plethora of research works then study how to improve the efficiency of the IM problem. Most algorithms rely on heuristics to identify those highly influential nodes but fail to provide the desired approximation guarantee.

To tackle this challenging issue, Borgs et al. [9] make a theoretical breakthrough that reduces the time complexity to $O(k(m + n) \log^2 n/\epsilon^3)$, which is almost linear to the graph size, while still providing $(1 - 1/e - \epsilon)$-approximation under the Independent Cascade model. They further prove a lower bound $\Omega(m + n)$ for the expected running time on general graphs under IC model. The key idea of their proposed solution is to generate a sufficiently large number of random *reverse reachable (RR) sets*, and then apply the greedy algorithm to select the k nodes. A line of follow-up research work then focuses on how to reduce the number of random reverse reachable sets to achieve better efficiency while providing the same approximation guarantee. The representatives include [42, 46–48]. Tang et al. [48] present *TIM/TIM+*, which reduces the time complexity to $O(k(m + n)\epsilon^{-2} \log n)$, and further show that the idea of reverse reachable sets can be applied to both IC and LT model. Later, Tang et al. [47] propose *IMM*, Nguyen et al. [42] develop *SSA* and *D-SSA*, and Tang et al. propose *OPIM-C* [46] to further improve the empirical efficiency by reducing the number of random RR sets generated without improving the time complexity. This line of *RR-set-based* solutions is shown to provide superb efficiency on large scale social networks under several popular cascade models. For instance, on Twitter network with 1.5 billion edges, OPIM-C can return an approximate answer within 10 seconds. However, these IM algorithms, using RR set as the backbone, suffer from scalability issues in high influence networks as evidenced by existing empirical studies [7]. How to tackle this challenge is still an open problem.

Motivated by this, in this paper, we present an in-depth study on the random RR set generation, the key phase for all existing RR-set-based solutions. Instead of trying to reduce the number of RR sets, we consider from a totally different perspective, by reducing the computational cost for generating a random RR set. We improve the efficiency of RR set generation by effective subset sampling and show that our new RR set generation algorithm improves over the existing RR set generation algorithm by up to an order of magnitude. With the new algorithm, we show that the expected running time of existing IM algorithms under IC model can be improved to

$O(k \cdot n \log n/\epsilon^2)$[1], when for any node v, the total weight of its incoming edges is no larger than a constant. We further show that without modifying the existing RR set generation algorithm under LT model, the time complexity can be improved to $O(k \cdot n \log n/\epsilon^2)$ as well.

Moreover, in high influence networks, the size of a random RR set tends to be extremely large, and it takes prohibitive computational and memory costs. For instance, in the pandemic, messages about COVID-19 are easily get propagated across the social network, resulting in a phenomena called Infodemic. In such scenarios, even if we apply our new algorithm to generate the random RR sets, it is still too expensive since the size of a random RR set is too large. To remedy this deficiency, we propose a non-trivial two-phase solution that significantly reduces the average size of random RR sets, making our solution practical for high influence networks. The main idea is that we first select a set B of b nodes as the seeds and then select the remaining $k - b$ nodes. When we select the remaining $k - b$ nodes, the RR set generation process can immediately stop when any node in B is reached. Thus, the average size of the random RR sets can be reduced. The main challenge is how to retain the approximation guarantee with this idea. We show that our proposed solution still provides the same theoretical result as existing solutions. Experimental results demonstrate that with our solution, the average size of random RR sets can be reduced by up to 700x. Our solution is further up to two orders of magnitude faster than alternatives.

Though the RR-set-based solution is highly efficient to find out a seed set for the IM problem, unfortunately it provides limited propagating information. For instance, given a seed set obtained by RR-set-based solutions, we have no idea about which nodes will be influenced with a probability higher than a certain threshold. Such a query is sensible on its own. In viral marketing, after providing products to influential users, a company has the motivation to further distribute coupons to those who are more likely to purchase the products, in hope of improving its sales performance. To tackle this issue, if we still insist to apply the RR-set-based solution, then we need to generate a large number of RR sets from each target node, which incurs prohibitive computational costs. However, another solution is to conduct many simulations starting from the obtained seed set S, each of which follows the definition of the influence propagation process. Then for node v, the fraction of the simulations in which v is finally activated is an estimator for the probability that v is influenced with respect to seed set S. In contrast to the RR-set-based approach, such simulations are referred to as *forward propagations*. From the aforementioned example, it is showed that the forward propagation could be a powerful tool in some scenarios, deserving our full attention. In this paper, we develop non-trivial results about this problem.

This manuscript is a journal extension to our previous conference paper [26]. We summarize the main differences from our conference version as follows:

- For general IC model where the weights are skewed, the index-based solution for the subset sampling problem might suffer from cache efficiency issues as it requires a two-dimensional index structure. In Section 4, we propose a sampling algorithm, dubbed as SKIP, which achieves the optimal time complexity of the index-free sorted subset sampling problem [11]. To our best knowledge, this is the first index-free algorithm that achieves the optimal time complexity. Our experimental results demonstrate that it further achieves better practical performance than existing solutions.
- In Section 6.1, we prove that with a constraint that the total weight of the incoming edges for any node $v \in V$ is bounded by a constant number, the cost of a forward propagation with our subset sampling technique is bounded by $O(\mathsf{I}_{\mathsf{C}}(S))$, where $\mathsf{I}_{\mathsf{C}}(S)$ is the expected influence of the seed set S. It is optimal since it costs at least $O(1)$ to sample an influenced node. Furthermore, experimental evaluations show its practical efficiency.

[1]The lower bound in [9] only applies to general IC model.

Table 1. Frequently used notations.

Notation	Description
$G(V, E)$	a social network with node set V and edge set E
n, m	$n = \|V\|$, and $m = \|E\|$
$IN(v), OUT(v)$	the set of in-neighbours and out-neighbours of node v, respectively
$d_{in}(v), d_{out}(v)$	$d_{in}(v) = \|IN(v)\|$, and $d_{out}(v) = \|OUT(v)\|$, respectively
d_{avg}	the average degree of an undirected graph
$\mathsf{I}_{\mathsf{C}}(S)$	the expected influence of S
OPT_k	the maximal $\mathsf{I}_{\mathsf{C}}(S)$ for any size-k seed set
S_k^o	an optimal seed set with $\mathsf{I}_{\mathsf{C}}(S_k^o) = OPT_k$
S_k^*	the size-k seed set returned by a certain algorithm
R	a random RR set
$\mathcal{R}$	a set of random RR sets, that is, $\mathcal{R} = \{R_1, R_2, \ldots\}$
$\Lambda_{\mathcal{R}}(S)$	the coverage of a seed set S with respect to $\mathcal{R}$
$\mathsf{I}_{\mathsf{C}}^-(S)$	a lower bound of the expected influence of S
$\mathsf{I}_{\mathsf{C}}^+(S_k^o)$	an upper bound of the expected influence of S_k^o

- Since a forward propagation is bounded by $O(\mathsf{I}_{\mathsf{C}}(S))$, if the given seed set has low expected influence, it is possible to reap some benefits from using the forward propagation approach when estimating influence of a given seed set. In Section 6.2, we first prove that it takes $O(d_{avg})$ to sample a random RR set on undirected graphs under WC model, where d_{avg} is the average degree. With this bound on hand, we further develop a heuristic condition to indicate when the forward propagation approach should be used. Experimental evaluations show that for those seed sets satisfying the proposed condition, the forward propagation estimator does provide a much better practical performance in terms of influence estimation.

The rest of this manuscript is organized as follows. Section 2 reviews the definition of the IM problem and existing solutions. Section 3 presents our SUBSIM framework for WC and Uniform IC model. Section 4 extends SUBSIM to general IC model with index-based and index-free solutions. Section 5 handles with the highly influential scenarios where existing RR-set-based solutions suffer from scalability issues. Section 6 focuses on the forward propagation issues. Section 7 reviews related work, and Section 8 shows the experimental evaluations.

2 PRELIMINARIES

2.1 Problem Definition

Let $G = (V, E)$ be a directed graph G with n nodes and m edges representing a social network where each node $v \in V$ represents a user and each edge $(u, v) \in E$ represents the relationship, e.g., friendship, between u and v. If $(u, v) \in E$, we say that u is the in-neighbor of v and v is the out-neighbor of u. Assume that each edge $e = (u, v)$ is associated with a weight $p(u, v) \in [0, 1]$, denoted as the *propagation probability*. Given a set S of nodes in G, we consider the following discrete-time stochastic cascade process C which applies to both the Independent Cascade and Linear Threshold model:

- At timestamp 0, all the nodes in set S are *activated* and the remaining nodes are *inactive*. A node activated will remain activated in subsequent timestamps.

- If a node is activated at timestamp i, it has a chance to activate its out-neighbors at timestamp $i+1$ according to some probability distribution (depending on the cascade model), after which it cannot activate any node.
- The influence propagation terminates when none of the activated nodes can activate other nodes.

Let $I_C(S)$ be the number of activated nodes in G for an instance C of above stochastic propagation C. We denote set S as the seed set and $I_C(S)$ as the influence of S in stochastic propagation instance C, and denote $\mathsf{I}_{\mathsf{C}}(S) = \mathsf{E}_{C\in\mathsf{C}}[I_C(S)]$ as the expected influence of S under the cascade process C. Table 1 lists the notations used frequently in this paper.

Definition 1 (Influence Maximization). *Given a graph G, a cascade model C, and an integer k, the influence maximization problem asks for a size-k seed set S_k with the largest expected influence, i.e., $S_k = \arg\max_{S':|S'|=k} \mathsf{I}_{\mathsf{C}}(S')$.*

Cascade Models. We focus on two widely adopted diffusion models: the *Independent Cascade (IC)* model and *Linear Threshold (LT)* model. Both models share the same discrete-time cascade process as mentioned in Section 2.1 and the main difference lies in how the inactive nodes get activated:

- IC model. Suppose node u gets activated at timestamp i, then u has a single chance to activate its inactive out-neighbor v with probability $p(u,v)$ at timestamp $i+1$.
- LT model. In the LT model, it assumes that for each node v: (i) the sum of the propagation probability of its incoming edges is no more than 1, and (ii) a probability λ_v is selected uniformly at random from $[0,1]$. If v is inactive at timestamp i, then it becomes activated at timestamp $i+1$ if and only if $\sum_{u\in A} p(u,v) \geq \lambda_v$, where A is the set of activated in-neighbor of v at timestamp i.

2.2 Existing Solutions

As mentioned in Section 1, most existing scalable IM methods utilize a sampling technique called *Reverse Influence Sampling (RIS)*, proposed by Borgs et al. [9]. This technique is based on the concept of *random reverse reachable (RR) set*. A random RR set R is constructed in two steps: *(i)* randomly select a node $v \in V$; *(ii)* reversely sample the set R of nodes that can activate v, such that for each node $u \in V$, the probability that it appears in R equals the probability that u can activate v. This set R is denoted as a *reverse reachable set* of v, and node v is the *target node* of the RR set R.

Under IC model, we can generate a random RR set as follows: Generate a directed graph g by removing each edge e with probability $1-p(e)$ independently, and denote $\mathcal{G}$ as the distribution of g. Given an instance g of distribution $\mathcal{G}$ and a node v, the reverse reachable set R for v in g is the set of nodes in g that can reach v. R is a random RR set if v is sampled uniformly at random from V. Intuitively, if a set S is highly influential, then there is a high chance that some nodes in S appear in the RR set of a randomly generated node v. Borgs et al. [9] establish the following connection between the expected influence of S and a random RR sample.

Lemma 1. *Let $S \subseteq V$ be a seed set and R be a random RR set generated with diffusion model C, then*

$$\mathsf{I}_{\mathsf{C}}[S] = n \cdot \Pr[S \cap R < \emptyset].$$

Lemma 1 indicates that we can estimate the expected influence of an arbitrary seed set S using random RR sets. We say S covers an RR set R if $S \cap R < \emptyset$. Assume that we generate a set $\mathcal{R}$ of random RR sets. Define the *coverage* $\Lambda_{\mathcal{R}}(S)$ of a seed S with respect to $\mathcal{R}$ as the number of RR sets in $\mathcal{R}$ that is covered by S. Then, $n \cdot \Lambda_{\mathcal{R}}(S)/|\mathcal{R}|$ provides an unbiased estimation of the expected influence of S.

Borgs et al.'s solution. With Lemma 1, Borgs et al. [9] propose a two-step method for IM. Firstly, a sufficiently large set $\mathcal{R}$ of random RR sets is generated. Given a node v and the set $\mathcal{R}$, define the

Algorithm 1: Max-Coverage-Greedy($\mathcal{R}, k$)

$S_k^* = \emptyset$;
for $i = 1$ *to* k **do**
  $v \leftarrow \arg\max_{v' \in V}(\Lambda_{\mathcal{R}}(S_k^* \cup \{v'\})) - \Lambda_{\mathcal{R}}(S_k^*)$;
  $S_k^* \leftarrow S_k^* \cup \{v\}$;
return S_k^*;

marginal coverage of v w.r.t a set S as:

$$\Lambda_{\mathcal{R}}(v|S) = \Lambda_{\mathcal{R}}(\{v\} \cup S) - \Lambda_{\mathcal{R}}(S).$$

Then, in the second phase of their solution, it simply applies the standard greedy algorithm as shown in Algorithm 1 that iteratively selects the node with the maximum marginal coverage with respect to the set of selected nodes in previous iterations. Denote this set as S_k^* and return S_k^* as the solution. Let $\hat{S}_k^o$ be the size-k seed set that covers the largest number of RR sets in $\mathcal{R}$ and S_k^o be the optimal seed that provides the highest expected influence. Then obviously, $\Lambda_{\mathcal{R}}(\hat{S}_k^o) \geq \Lambda_{\mathcal{R}}(S_k^o)$. Then, the greedy algorithm guarantees that:

$$\Lambda_{\mathcal{R}}(S_k^*) \geq (1 - 1/e)\Lambda_{\mathcal{R}}(\hat{S}_k^o) \geq (1 - 1/e)\Lambda_{\mathcal{R}}(S_k^o).$$

Borgs et al. show that S_k^* provides a $(1 - 1/e - \epsilon)$-approximate solution with probability at least $1 - 1/n$ if $O(k(m+n)\epsilon^{-3}\log^2 n)$ edges are examined in the RR set generation.

TIM+ and IMM. Tang et al. [48] present an improved algorithm *TIM+* , which runs in $O(k \cdot (n+m)\epsilon^{-2} \cdot \log n)$ time. The main idea is to use Chernoff bound to decide if the number of RR sets, instead of the number of edge examined, is sufficient to provide an approximation guarantee. Later, Tang et al. [47] present *IMM* that uses a martingale-based technique to allow the random RR set to have some weak dependencies without affecting the concentration bound. They apply below two martingale-based concentration bounds tailed for IM.

Lemma 2 ([47]). *Given a fixed number θ of random RR sets and a seed set S, for any $\lambda > 0$,*

$$\Pr\left[\Lambda_{\mathcal{R}}(S) - \mathsf{I}_{\mathsf{C}}(S) \cdot \frac{\theta}{n} \geq \lambda\right] \leq \exp\left(-\frac{\lambda^2}{2\mathsf{I}_{\mathsf{C}}(S) \cdot \frac{\theta}{n} + \frac{2}{3}\lambda}\right),$$

$$\Pr\left[\Lambda_{\mathcal{R}}(S) - \mathsf{I}_{\mathsf{C}}(S) \cdot \frac{\theta}{n} \leq -\lambda\right] \leq \exp\left(-\frac{\lambda^2}{2\mathsf{I}_{\mathsf{C}}(S) \cdot \frac{\theta}{n}}\right).$$

As shown in [47], *IMM* offers the same guarantee as that of *TIM+*, but gains better practical performance since it reduces the number of RR set samples and thus the query time.

SSA and D-SSA. All previous methods are pessimistic about the seed set selected in the greedy algorithm and thus apply the union bound on the possible $\binom{n}{k}$ size-k seeds for the case when the seed set selected does not provide an approximation guarantee. Thus, the final time complexity will depend on k and the larger k it is, the more RR sets are required to provide the approximation ratio. Nguyen et al. [42] propose *SSA* and *D-SSA* to alleviate the (empirical) dependency on k by being optimistic about the seeds selected by Algorithm 1 and then use a validation phase to verify if the chosen seed is good or not. They claim that they provide the same theoretical result as *IMM*, but Huang et al. [28] show that the theoretical analysis of *SSA* and *D-SSA* contains loopholes that invalidate the claimed time complexity and approximation guarantee. Huang et al. further present

Algorithm 2: RR set-Generation-IC(G)

```
Randomly sample a node v ∈ V and set R as {v};
Add v to queue Q and mark v as activated ;
while Q is not empty do
    Let u be the top element of Q. Pop it from Q;
    for each in-neighbor w of u do
        if w is inactivated and rand() ≤ p(w, u) then
            Add w to R;
            Add w queue Q and mark w as activated;
    return R;
```

SSA-Fix to reassure the $(1-1/e-\epsilon)$-approximation guarantee with $1-1/n$ probability and pinpoint that it is unclear how to provide efficiency and approximation guarantee for *D-SSA*. Nguyen et al. [39] further present *D-SSA-Fix* to restore the $(1-1/e-\epsilon)$-approximation guarantee, but the efficiency guarantee of *D-SSA-Fix* is still unclear, as pointed out in [28, 46].

SKIS. A *singular* RR set is an RR set which includes only one node (i.e., the target node of the RR set). Nguyen et al. [40] propose a sketching framework, SKIS, which uses the importance influence sampling method to generate a collection of only non-singular RR sets for influence estimation. However, such a strategy might not significantly improve the practical performance, since the generation of a singular RR set does not take much computational cost. To explain, it can stop immediately after all the incoming edges of the target node have been blocked, and thus the expected cost of a random singular RR set is only the average in-degree of the graph. Besides, when we apply the greedy algorithm to get a potential seed set, the SKIS-based solution has to utilize the max heap method so as to find out the node with the **largest marginal influence** which is float data type in each iteration. In contrast, for the RR-set-based solution, a more efficient implementation of inverted list and lazy update for the greedy algorithm can be used to find out the node with the **largest marginal coverage** which is integer data type. This issue also degrades the practical performance of the SKIS-based solution.

OPIM-C. The latest RR-set-based solution for IM is the *OPIM-C* algorithm [46]. *OPIM-C* shares a similar spirit as *SSA*/*D-SSA* in that they are both optimistic about the selected seed set by the greedy algorithm. In *OPIM-C*, they first sample a set $\mathcal{R}_1$ of RR sets to select the seed set S_k^* and derive the upper bound $\mathsf{I}_{\mathsf{C}}^{+}(S_k^o)$ of $\mathsf{I}_{\mathsf{C}}(S_k^o)$. Next, they sample another set $\mathcal{R}_2$ of random RR sets with $|\mathcal{R}_2| = |\mathcal{R}_1|$ and derive a lower bound $\mathsf{I}_{\mathsf{C}}^{-}(S_k^*)$ of $\mathsf{I}_{\mathsf{C}}(S_k^*)$. The algorithm terminates as soon as

$$\mathsf{I}_{\mathsf{C}}^{-}(S_k^*)/\mathsf{I}_{\mathsf{C}}^{+}(S_k^o) \;¡\; (1-1/e-\epsilon),$$

i.e., when the algorithm provides a $(1-1/e-\epsilon)$-approximate solution. In *OPIM-C*, the authors present strategies to provide a tighter upper bound $\mathsf{I}_{\mathsf{C}}^{+}(S_k^o)$ of $\mathsf{I}_{\mathsf{C}}(S_k^o)$. With tighter bounds, the number of RR set samples can be reduced, thus improving the running time. By applying Lemma 2, Tang et al.[46] derive the lower bound $\mathsf{I}_{\mathsf{C}}^{-}(S_k^*)$ as follows:

$$\mathsf{I}_{\mathsf{C}}^{-}(S_k^*) = \left(\sqrt{\Lambda_{\mathcal{R}_2}(S_k^*) + \frac{2\eta_l}{9}} - \sqrt{\frac{\eta_l}{2}}\right)^2 - \frac{\eta_l}{18} \cdot \frac{n}{\theta_2}, \tag{1}$$

where $\eta_l = \ln(1/\delta_l)$ and δ_l is the probability that the above lower bound fails. By applying Lemma 2, the upper bound $\mathsf{I}^+_{\mathsf{C}}(S^o_k)$ is given as follows:

$$\mathsf{I}^+_{\mathsf{C}}(S^o_k) = \left(\sqrt{\Lambda^u_{\mathcal{R}_1}(S^o_k) + \frac{\eta_u}{2}} + \sqrt{\frac{\eta_u}{2}}\right)^2 \cdot \frac{n}{\theta_1}, \tag{2}$$

where $\eta_u = \ln(1/\delta_u)$ and δ_u is the probability that the above upper bound fails; $\Lambda^u_{\mathcal{R}_1}(S^o_k)$, an upper bound of the coverage of S^o_k with respect to $\mathcal{R}_1$, is derived as follows. Though the optimal seed set S^o_k is unknown, the upper bound $\Lambda^u_{\mathcal{R}_1}(S^o_k)$ can be obtained from the construction of S^*_k due to the submodular property of coverage function $\Lambda(\cdot)$. Let S^*_i be the set that contains the first i nodes selected by running the greedy algorithm and $maxMC(S^*_i, l)$ be the set of l nodes with the l largest marginal coverage in $\mathcal{R}_1$ with respect to S^*_i. Then,

$$\Lambda^u_{\mathcal{R}_1}(S^o_k) = \min_{0 \le i \le k} \left\{ \Lambda_{\mathcal{R}_1}(S^*_i) + \sum_{v \in maxMC(S^*_i, k)} \Lambda_{\mathcal{R}_1}(v|S^*_i) \right\}.$$

2.3 RR Set Generation

Most of the above solutions (except SKIS) focus on reducing the number of random RR sets and are identical in how random RR sets are generated. Instead of first generating the graph g by flipping a coin for each edge that incurs $O(m)$ cost, the existing RR set generation algorithm for IC model, as shown in Algorithm 2, starts a traversal from v following the reverse directions of its edges. Such an approach only examines the in-coming edges of nodes in R, and thus significantly reduces the running cost for generating an RR set. We refer readers to [46] on how to generate a random RR set under LT model. According to [9, 48], a random RR set can be constructed in $O(\frac{m}{n} \cdot \mathsf{I}_{\mathsf{C}}(v^*))$ expected time, where $\mathsf{I}_{\mathsf{C}}(v^*)$ is the expected influence of a node v^* sampled from V where each v is sampled with a probability of $d_{in}(v)/m$.

Among existing solutions, SKIS is the only work which improves the efficiency of the IM problem by optimizing the RR set generation phase: it pre-computes the influence contribution of the singular RR sets for each node v in advance, and thus when estimating the influence of a seed set, what SKIS needs to do is to sample the non-singular RR sets via importance sampling, accelerating the RR set generation. However, the generation for the non-singular RR sets is still the same as existing solutions, limiting its effectiveness. In addition, as we mentioned in Section 2.2, SKIS disallows the efficient implementation of the greedy algorithm, which also degrades the overall performance of the SKIS-based solutions.

In this paper, we will present a theoretical study on the RR set generation phase to improve the efficiency of the IM problem. Our proposed solution is from an orthogonal perspective against SKIS and it can be applied to any RR set, no matter whether the RR set is non-singular or not.

3 SUBSIM

This section presents our SUBSIM (Subset Sampling with Influence Maximization) framework for IM. We present an efficient RR set generation scheme under WC and Uniform IC model in Section 3.1 and show improved theoretical results on IM algorithms with this new scheme in Section 3.2. We extend our SUBSIM to general IC model in Section 4.

3.1 A New RR set Generation Scheme

In the existing RR set generation algorithm (Algorithm 2), an expensive step is that when a node gets activated, it examines all of its in-neighbors and tries to activate each of them once (Algorithm 2 Line 6). In particular, it generates a random number for each incoming edge to determine if each

Algorithm 3: SUBSIM(G)

```
1  Randomly sample a node v ∈ V and set R as {v};
2  Add v to queue Q and mark v as activated;
3  while Q is not empty do
4      Let u be the top element of Q. Pop it from Q;
5      Let u[i] (i = 1, 2, ...) be the ith in-neighbor of u;
6      p ← 1/d_in(u) under WC;
7      i ← ⌈log(rand())/log(1 − p)⌉;
8      while i ≤ d_in(u) do
9          w ← u[i];
10         if w is not activated then
11             Add w to R;
12             Add w to queue Q and mark w as activated;
13         i += ⌈log(rand())/log(1 − p)⌉;
14     return R;
```

of its in-coming neighbors will be activated or not. That is actually why the time complexity of existing IM algorithms depends on the average degree, i.e., m/n. With subset sampling, we show that the expected cost to sample an edge e under IC model can be reduced to $O(p(e))$.

Connection with Subset Sampling. We make a connection between *subset sampling* with the selection of in-neighbors. Given a set $S = \{x_1, x_2, x_3, \cdots, x_h\}$ of h elements, and each with a weight $0 \leq p(x_i) \leq 1$. Denote μ as the sum of all the weights, i.e., $\mu = \sum_{i=1}^{h} p(x_i)$. The independent subset sampling problem asks to sample a random subset X such that each element x_i in S will be independently added to set R with probability $p(x_i)$. The problem of activating the in-neighbors of a node v can be directly mapped to the subset sampling problem. We first consider the case where all weights are equal and denote this weight as p, which covers the scenarios of WC, where the weights of the incoming edges of the same node v are $1/d_{in}(v)$, and Uniform IC where all edges have the same weight p.

When the probabilities are the same, the subset sampling can be effectively solved with geometric distribution sampling. In particular, we are interested in the event that we successfully sample the first element from S after X trials. The probability distribution of X follows the geometric distribution $\mathrm{G}(p)$ and the probability is given as follows:

$$\Pr[X = i] = (1-p)^{i-1} \cdot p,$$

where $i = 1, 2, 3, \cdots$. If $i \; j \; h$, it indicates that no element is sampled from set S. Notice that in distribution $\mathrm{G}(p)$, all trials are assumed to be independent, and therefore it still guarantees that the sampling of each in-neighbor should be independent. This leads to our RR set generation algorithm for WC and Uniform IC model as shown in Algorithm 3. The main difference from Algorithm 2 is Lines 7 and 13, where the algorithm jumps to skip nodes that are not sampled, saving computational costs. Assume that an $h' \leq h$ is sampled from distribution $\mathrm{G}(p)$, the first $h'-1$ elements are skipped and it directly jumps to the h'-position, sampling element $x_{h'}$. Then, it continues to sample the first element from the remaining $h - h'$ nodes. This process is repeated until the sampled h' is larger than the number of remaining elements. Note that there exist constant time solutions [31] to sample from distribution $\mathrm{G}(p)$: Given a U generated uniformly at random from $(0, 1)$, we can

sample h' from $\mathrm{G}(p)$ as

$$h' = \lceil \log U / \log(1-p) \rceil.$$

To explain, $h' = i$ if and only if $U \in [(1-p)^i, (1-p)^{i-1})$, which has a probability of $(1-p)^{i-1} - (1-p)^i = (1-p)^{i-1} \cdot p$, i.e., following distribution $\mathrm{G}(p)$. Therefore, the expected cost of the sampling phase only depends on the number of times we do geometric sampling.

Lemma 3. *Given a set S of h elements each to be sampled independently with probability p, then the expected cost for sampling a subset R is $O(1+\mu)$, where $\mu = h \cdot p$.*

Proof. Based on the new sampling strategy, each record is sampled with probability p. For all h records, the probability to sample each edge is $h \cdot p$. Since we need to generate at least one random number, the cost is $O(1 + h \cdot p) = O(1+\mu)$.

Given above results, new bounds can be derived for IM.

3.2 Influence Maximization: A New Bound

With SUBSIM for the RR set generation, we show that the time complexity of existing IM algorithms can be tightened. We first analyze the running cost of SUBSIM for the RR set generation. The running cost can be bounded by *the number of edges examined* during the RR set generation. Denote $\theta(x)$ as a function depending only on x, and then we give the following lemma to bound the running cost of SUBSIM.

Lemma 4. *If θ is a concave function and for any node v, $\sum_{(u,v)\in E} p(u,v) \le \theta(d_{in}(v))$, the cost to generate a random RR set under WC and Uniform IC model can be bounded by $\theta(m/n) \cdot \mathsf{I_C}(\{v^*\})$, where v^* is sampled from a distribution where node v has $\frac{\theta(d_{in}(v))}{\sum_{w\in V}\theta(d_{in}(w))}$ probability to be sampled.*

Proof. We first consider the cost to generate an RR set with a fixed target node v. Let $\Pr[v \xrightarrow{R} u]$ denote the probability that u is included in the RR set, i.e., u is activated in the reverse stochastic traverse from v; let $\Pr[v \xrightarrow{R} (w,u)]$ indicate the probability that (w,u) is examined. Then, (w,u) is examined if and only if these two events both happen: (a) u is activated in the reverse stochastic traverse from v; (b) edge (w,u) is selected by geometric sampling. With the fact that the expected cost to examine (w,u) is actually $p(w,u)$ under geometric sampling, we can derive the expected cost $\mathsf{E}[v \xrightarrow{R} (w,u)]$ of edge (w,u) to be:

$$\mathsf{E}[v \xrightarrow{R} (w,u)] = \Pr[v \xrightarrow{R} (w,u)] = \Pr[v \xrightarrow{R} u] \cdot p(w,u).$$

The expected cost to generate an RR set with respect to target node v, denoted as $\mathsf{E}[C_{R(v)}]$, is:

$$\begin{aligned}
\mathsf{E}[C_{R(v)}] &= \mathsf{E}\left[\sum_{(w,u)\in E} v \xrightarrow{R} (w,u)\right] = \sum_{(w,u)\in E} \mathsf{E}\left[v \xrightarrow{R} (w,u)\right] \quad \text{(by linearity of expectation)} \\
&= \sum_{(w,u)\in E} \Pr[v \xrightarrow{R} (w,u)] = \sum_{(w,u)\in E} \Pr[v \xrightarrow{R} u] \cdot p(w,u) \\
&= \sum_{u\in V} \Pr[v \xrightarrow{R} u] \cdot \sum_{(w,u)\in E} p(w,u) \le \sum_{u\in V} \theta(d_{in}(u)) \cdot \Pr[v \xrightarrow{R} u].
\end{aligned}$$

Now consider the cost of a random RR set, denoted as E_R.

$$E_R = \frac{1}{n} \sum_{v\in V} \mathsf{E}[C_{R(v)}] \le \frac{1}{n} \sum_{v\in V} \sum_{u\in V} \theta(d_{in}(u)) \cdot \Pr[v \xrightarrow{R} u].$$

Further observe that $\Pr[v \xrightarrow{R} u]$ is equal to the probability that u can influence v, denoted as $\Pr[u \to v]$. Let $\theta(V) = \sum_{w \in V} \theta(d_{in}(w))$. Then, we can derive that:

$$\begin{aligned} E_R &\leq \frac{\theta(V)}{n} \sum_{v \in V} \sum_{u \in V} \frac{\theta(d_{in}(u))}{\theta(V)} \cdot \Pr[u \to v] \\ &= \frac{\theta(V)}{n} \sum_{u \in V} \frac{\theta(d_{in}(u))}{\theta(V)} \sum_{v \in V} \Pr[u \to v]. \end{aligned}$$

Notice that $\sum_{v \in V} \Pr[u \to v]$ indicates the expected influence of node u. Further let node v^* be a node sampled from a distribution where each node v is sampled with probability $\frac{\theta(d_{in}(v))}{\theta(V)}$. We can further derive that:

$$\begin{aligned} E_R &\leq \frac{\theta(V)}{n} \sum_{u \in V} \frac{\theta(d_{in}(u))}{\theta(V)} \mathsf{I}_C(\{u\}) \\ &= \frac{\theta(V)}{n} \cdot \mathsf{I}_C(\{v^*\}) \leq \theta(m/n) \cdot \mathsf{I}_C(\{v^*\}) \end{aligned} \tag{3}$$

where the last inequality is due to the concavity of the function θ. This finishes the proof.

Theorem 1. *If θ is a concave function and for any node v, $\sum_{(u,v) \in E} p(u,v) \leq \theta(d_{in}(v))$, the time complexity of IM algorithms under WC and Uniform IC model to provide a $(1 - 1/e - \epsilon)$-approximate solution with $1 - 1/n$ probability can be bounded by $O(k \cdot \theta(m/n) \cdot n \cdot \log n/\epsilon^2)$.*

Proof. Note from [47] that, the number of RR sets can be bounded by $O(\frac{k \cdot n \cdot \log n}{OPT_k \cdot \epsilon^2})$, where OPT_k is the largest expected influence among all seed sets with size no more than k. Then, since $\mathsf{I}_C(\{v^*\}) \leq OPT_k$, we know that $E_R = O(\theta(m/n) \cdot OPT_k)$. Combining them together, we have:

$$O(\frac{k \cdot n \cdot \log n}{OPT_k \cdot \epsilon^2} \cdot E_R) = O(k \cdot \theta(m/n) \cdot n \cdot \log n/\epsilon^2).$$

This finishes the proof.

With Theorem 1, we immediately have the following conclusions for three useful cases.

- **Case 1: $\theta(x) = O(1)$.** WC model falls into this case, and the time complexity becomes $O(k \cdot n \cdot \log n/\epsilon^2)$, improving over existing solutions by $O(m/n)$.
- **Case 2: $\theta(x) = O(\log(x))$.** The time complexity becomes $O(k \cdot \log(m/n) \cdot n \cdot \log n/\epsilon^2)$, which still improves over existing solutions by $O(m/n/\log(m/n))$.
- **Case 3: $\theta(x) = O(p \cdot x)$.** Uniform IC falls into this case, and the time complexity becomes $O(p \cdot k \cdot (m+n) \cdot \log n/\epsilon^2)$, improving over existing solutions by $O(p)$.

Extensions to LT model. Notice that under LT model, the cost to sample an edge is also proportional to its weight [46, 47], and it naturally holds that $\sum_{u \in IN(v)} p(u,v) \leq 1$, where $IN(v)$ is the set of the in-neighbors of v. By following the proof of Lemma 4 and Theorem 1, it can be derived that the time complexity of existing IM algorithms under LT model can be reduced to $O(k \cdot n \cdot \log n \cdot \epsilon^{-2})$.

4 EXTENSION TO GENERAL IC MODEL

In Section 3.1, we only discuss WC and Uniform IC, where the weights of the incoming edges of the same node are equal. However, in practice, the weights might be skewed, e.g., following exponential distribution, Weibull distribution [47], or by learning from data [22, 23]. In this section, we discuss how to handle general IC model.

The rest of this section is organized as follows. In Section 4.1, we propose an algorithm, Index-based Subset Sampling (ISS), which achieves the same time complexity as the WC and Uniform IC

Algorithm 4: ISS

Input: Buckets $B_k (0 \le k \le L)$ and probability table T;
Output: A subset S sampled from probability distribution $\{p_i\}$;
$K \leftarrow \emptyset$;
$k \leftarrow$ aliasSampling($T[0, :]$);
while $k \le L$ **do**
    $K \leftarrow K \cup \{k\}$;
    $k \leftarrow$ aliasSampling($T[k+1, :]$); • the $(k+1)$-th row of T
$S \leftarrow \emptyset$;
for $k \in K$ **do**
    $p \leftarrow 2^{-k}$;
    $r \leftarrow rand((1-p)^{|B_k|}, 1.0)$; • random number from $((1-p)^{|B_k|}, 1.0)$
    $i \leftarrow \lceil \log(r)/\log(1-p) \rceil$; • $i \in \{1, \ldots, |B_k|\}$
    do
        if *rand()* Ÿ p_i/p **then**
            $S \leftarrow S \cup \{x_i\}$;
        $i \leftarrow i + \lceil \log(rand())/\log(1-p) \rceil$; • geometric distribution sampling
    while i Ÿ= $|B_k|$;
return S;

***Preprocessing*:**
$L \leftarrow \lceil \log_2 h \rceil$;
Divide the set X of elements x_i into L buckets: $B_k = \{p_i | 2^{-k} \ge p_i \ge 2^{-k-1}|\}$ where $0 \le k \le L-1$ and $B_L = \{p_i | 2^{-L} \ge p_i\}$;
Create the probability table T where $T[i, j]$ stores the probability that B_i is the current sample bucket and B_j is the next bucket that will be sampled;

cases, at a cost of additional preprocessing indices. In Sections 4.2 and 4.3, we present two index-free online solutions for the sorted subset sampling problem. In particular, in Section 4.2 we review a bucket-based solution proposed in [11]. For ease of explanation, we name it *BUCKET*. In Section 4.3 we present the algorithm *SKIP*, of which the time complexity achieves the asymptotically tight bound of the sorted subset sampling problem.

4.1 ISS Algorithm

In this subsection, we provide an index-based solution ISS to handle general IC model. We still map the selection of in-neighbors to subset sampling and have the following lemma from [11] to bound its expected cost.

Lemma 5. *Given a set $S = \{x_1, x_2, \cdots, x_h\}$ of h elements where x_i is independently sampled with p_i probability, the expected running time to sample a subset X can be bounded by $O(1+\mu)$ with $O(h)$ preprocessing time, where $\mu = \sum_{i=1}^{h} p_i$.*

The main idea of Lemma 5 is to first divide the probability into different buckets such that p_i falls into a bucket B_k if $2^{-k} \ge p_i \ge 2^{-k-1}$ (resp. $2^{-k} \ge p_i$), where $0 \le k \le \lceil \log_2 h \rceil - 1$ (resp. $k = \lceil \log_2 h \rceil$). Then, in each bucket B_k, we first treat all probability in the bucket to be 2^{-k}, and then

Algorithm 5: BUCKET

Input: A set X of elements $\{x_i\}$ with probability $\{p_i\}$ in decending order.
Output: A subset S sampled from probability distribution $\{p_i\}$.
$S \leftarrow \emptyset$; $h \leftarrow |X|$; $L \leftarrow \lfloor \log_\beta h \rfloor$;
for $k = 0$ *to* L **do**
    $p \leftarrow p_{\beta^k}$;
    $i \leftarrow \beta^k + \lceil \log(rand())/\log(1-p) \rceil - 1$;
    while $i \leqslant \min(\beta^{k+1}, h+1)$ **do**
        if *rand()* $\leqslant p_i/p$ **then**
            $S \leftarrow S \cup \{x_i\}$;
        $i \leftarrow i + \lceil \log(rand())/\log(1-p) \rceil$;
return S;

apply geometric sampling to sample a position h'. When $h' \leq |B_k|$, we skip $h'-1$ elements (like Algorithm 3) and try to sample the h'-th element in B_k. However, we further generate a random variable U and successfully sample the h'-th element only if U is no larger than $p_{h'}/2^{-k}$ where $p_{h'}$ is the probability of the h'-th element in B_k. By this strategy, the h'-th element is still guaranteed to be sampled with $2^{-k} \cdot p_{h'}/2^{-k} = p_{h'}$ probability. For each bucket, the expected sampling cost increases by at most twice (For the last bucket, it increases to at most $1/h$). Therefore, the total expected cost can be bounded by $O(1+\mu+\log h)$, where the $\log h$ term comes from sampling in $O(\log h)$ buckets.

Next, we show how to further reduce the $\log h$ term. Firstly, we calculate the probability to do at least one geometric sampling from each bucket. Since each bucket B_k includes at least one geometric sampling can be calculated as $p'_k = 1-(1-2^{-k})^{|B_k|}$. This can be calculated with $O(\log h)$ time as it includes $O(\log h)$ bucket. Then, the problem becomes a new subset set sampling problem, where we are independently sampling each bucket B_k with probability p'_k. To avoid testing for each bucket, an $L \times L$ table can be maintained where $T[i,j]$ records the probability that B_i is the current sampled bucket and B_j ($i \leqslant j$) is the next bucket after i that will be sampled. We can calculate the probability of table T in $O(L^2) = O(\log^2 h)$ time. Also, given a current position i, we can sample according to the probability $T[i,i+1], T[i,i+2], \cdots T[i,h]$ in $O(1)$ time using alias sampling [50]. Then, we can sample the buckets first with $O(1+\mu)$ time, and sample within each bucket next. The total cost to sample in each bucket can be bounded by $O(1+\mu)$ time. Hence, the total cost to sample a subset X from set S can be bounded by $O(1+\mu)$. Algorithm 4 shows the pseudocode of the above mentioned algorithm.

By Lemma 5 and Theorem 1, we have the following theorem.

Theorem 2. *If θ is a concave function and for any node v, $\sum_{(u,v)\in E} p(u,v) \leq \theta(d_{in}(v))$, the time complexity of IM algorithms under general IC model can be bounded by $O(k \cdot \theta(m/n) \cdot n \cdot \log n/\epsilon^2)$ so as to provide a $(1-1/e-\epsilon)$-approximate solution with $1-1/n$ probability.*

However, to achieve $O(1+\mu)$ expected running time, ISS requires the two-dimensional index structure, which is quite complicated. It may suffer from cache efficiency issues, especially for those low-degree nodes, thus giving undesirable practical performance. To tackle this deficiency, we present index-free subset sampling solutions in the following subsections, which only require the incoming edges to be sorted in descending order of their probabilities.

Algorithm 6: SKIP

Input: A set X of elements x_i with probability $\{p_i\}$ in descending order
Output: A subset S sampled from probability distribution $\{p_i\}$
$S \leftarrow \emptyset$;
$i \leftarrow 1$;
while $i \leq h$ **do**
  $j \leftarrow i + \lfloor \log(rand())/\log(1 - p_i) \rfloor$;
  if j ¡ h **then**
    return S;
  if $rand() \leq p_j/p_i$ **then**
    $S \leftarrow S \cup \{x_j\}$;
  $i \leftarrow j + 1$;

4.2 BUCKET Algorithm

In this subsection, we review the algorithm BUCKET (Algorithm 5) proposed in [11], and show its time complexity if we are granted additional prior knowledge. This outcome will play a key role on deriving the time complexity of our algorithm SKIP. That is why we include it here.

According to [11], if the elements $x_1, x_2, \cdots, x_h$ of set X are sorted in descending order of their probabilities ($p_1, p_2, \cdots, p_h$, respectively), subset sampling could be efficiently conducted based on a bucketing technique. Specfically, we do bucketing by their sorted positions such that elements whose positions fall into the range $[\beta^k, \beta^{k+1})$ belong to bucket B_k. Then, we exploit geometric distribution sampling with rejection (similar with ISS) to sample all the buckets. Note from [11] that, the total expected cost to sample from all buckets is bound to $O(1 + \beta\mu + \log_\beta h)$, where the $\log_\beta h$ term comes from the number of buckets. The asymptotically tight bound of the sorted subset sampling problem is as follows:

Lemma 6. *[11] There exists an online algorithm for sorted subset sampling with expected time $O(\mu)$ if $\mu \geq \frac{1}{2}\log h$, and $O\left(1 + \frac{\log h}{\log((\log h)/\mu)}\right)$ otherwise. The bound is asymptotically tight for any fixed μ.*

If we have knowledge of the value $\mu = \sum_{i=1}^{h} p_i$, it is possible for BUCKET to achieve the asymptotically tight bound, by assigning β with a suitable value. If $\mu \geq \frac{1}{2}\log h$, the setting $\beta = 2$ yields a time complexity of $O(1 + \beta\mu + \log_\beta h) = O(\mu)$. Otherwise, we minimize the time complexity by solving the following equation,

$$1 + \beta\mu = \log_\beta h, \tag{4}$$

It turn out to be $\beta = \Theta\left(\frac{\log h}{\mu}/\log\frac{\log h}{\mu}\right)$. Then, by plugging in β, we obtain a time complexity of $O(1 + \beta\mu + \log_\beta h) = O\left(1 + \frac{\log h}{\log((\log h)/\mu)}\right)$.

Hence, the actual time complexity of BUCKET depends on β, which should be decided according to the probability summation μ of the input sequence. However, the requirement of prior knowledge of μ is not always met, imposing much limitation on its application scenarios. If there exists no prior knowledge of μ, one common implementation is $\beta = 2$. As a result, BUCKET fails to achieve the asymptotically tight time complexity if μ is not $\Theta(\log h)$.

4.3 SKIP Algorithm

In this subsection, we present our solution SKIP, of which the time complexity is asymptotically tight. The name *SKIP* comes from the activity *Stone Skipping*, since our algorithm acts in a similar manner: landing and immediately bouncing at a nearby place.

Its pseudocode is shown in Algorithm 6. Similar with BUCKET, it also utilize the geometric distribution sampling (Line 6) to get the potentially sampled elements, and then use rejection (Lines 9-10) to guarantee the correctness of sampling probability. The crux of SKIP is that it keeps updating the success probability of geometric distribution with the probability of the currently visiting element. To explain, if we arrive at position $i-1$ in the current sampling trial (no matter whether the element x_{i-1} are finally sampled or rejected), then the success probability of the next sampling trial is p_i, the probability of the element x_i (see p_i in Line 6). Intuitively, this strategy is more efficient since SKIP adjusts the success probability in a timely fashion: following the actual probability distribution of the input sequence. In contrast, BUCKET fixes the probability of the first element as the success probability, and then goes through the whole bucket.

Correctness of SKIP. Next, we prove the following lemma to show the correctness of SKIP; that is, the output of SKIP follows the definition of sorted subset sampling.

Theorem 3. *Let Y be an output random variable of SKIP. Then given any subset S, it holds that*

$$\Pr[Y = S] = \left(\prod_{i \notin S} (1 - p_i)\right) \cdot \left(\prod_{j \in S} p_j\right) \tag{5}$$

where the right-hand side is the definition of sorted subset sampling.

Proof. Let $S = \{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$ with $j_1 < j_2 < \cdots < j_m$, where $m = |S|$. Then we can rewrite the probability expression (5) as the following formatting:

$$\Pr[Y = S] = p_{j_1} \left(\prod_{i=1}^{j_1-1} (1 - p_i)\right) \cdot p_{j_2} \left(\prod_{i=j_1+1}^{j_2-1} (1 - p_i)\right) \cdots p_{j_m} \left(\prod_{i=j_{m-1}+1}^{j_m-1} (1 - p_i)\right) \cdot \left(\prod_{i=j_m+1}^{n} (1 - p_i)\right). \tag{6}$$

Let Z_is be random variables of the **indices** of the sampled elements by SKIP, where $i = 1, 2, \ldots, m$. Besides, in order to analyze the sampling behavior from x_{j_m} to x_h, we append infinite dummy elements to the end of the input sequence, each with the same sampling probability p_h (i.e., $p_h = p_{h+1} = p_{h+2} = \cdots$). The SKIP algorithm is also slightly modified: when j exceeds the last element x_h of X (see Line 7), keep sampling until we obtain the $(m+1)$-th successfully sampled element $x_{j_{m+1}}$ (clearly $x_{j_{m+1}}$ is a dummy element). Let Z_{m+1} be a random variable of the **index** of this dummy element. If we can prove the conditional probability distribution of Z_i with respect to Z_{i-1} ($i = 1, 2, \ldots, m$) satisfies

$$\Pr[Z_i = j_i | Z_{i-1} = j_{i-1}] = \prod_{i=j_{i-1}+1}^{j_i-1} (1 - p_i) \cdot p_{j_i}, \tag{7}$$

and the conditional probability of the event $Z_{m+1} > h$ given $Z_m = j_m$ satisfies

$$\Pr[Z_{m+1} > h | Z_m = j_m] = \prod_{i=j_m+1}^{n} (1 - p_i), \tag{8}$$

then we can prove (6) as follows:

$$\begin{aligned}\Pr[Y = S] &= \Pr[Z_1 = j_1, Z_2 = j_2, \ldots, Z_m = j_m, Z_{m+1} \neq h] \\ &= \Pr[Z_1 = j_1] \cdot \Pr[Z_2 = j_2 | Z_1 = j_1] \cdots \Pr[Z_m = j_m | Z_{m-1} = j_{m-1}] \cdot \Pr[Z_{m+1} \neq h | Z_m = j_m] \\ &= \text{the r.h.s. of (6)}.\end{aligned}$$

Next, we try to prove the conditional probability distribution (7) by induction. We only give the proof for the case of $i = 1$; that is,

$$\Pr[Z_1 = j] = p_j \prod_{i=1}^{j-1} (1 - p_i), \tag{9}$$

where we replace j_1 with j for simplicity. It is easy to extend other values of i.

For $j = 1$, it implies that the random number $U = rand()$ (Line 6 in Algorithm 6) falls into the range $[1 - p_1, 1)$. This event happens with a probability p_1.

Suppose that the conclusion is still true for $j = k$. Note that the sampling trial which successfully samples x_k could start from position $1, 2, \ldots, k$ (that is, the i in Line 6 could be $1, 2, \ldots, k$). Let E_i^k be the event that the successful sampling of x_k is from position i, where $i = 1, 2, \ldots, k$. Obviously, any two E_i^ks are mutually exclusive. Besides, the event $Z_1 = k$ is the union of all E_i^ks. Then we rewrite the probability of $Z_1 = k$ as follows:

$$\begin{aligned}\Pr[Z_1 = k] &= \Pr[E_1^k] + \Pr[E_2^k] + \cdots + \Pr[E_k^k] \\ &= \alpha_1 (1 - p_1)^{k-1} p_1 \cdot \frac{p_k}{p_1} + \alpha_2 (1 - p_2)^{k-2} p_2 \cdot \frac{p_k}{p_2} + \cdots + \alpha_k p_k \\ &= \left(\sum_{i=1}^{k} \alpha_i (1 - p_i)^{k-i} \right) \cdot p_k = p_k \prod_{i=1}^{k-1} (1 - p_i) \quad \text{(from assumption)}\end{aligned}$$

where α_i is the probability that SKIP has arrived at position $(i - 1)$ but no element is successfully sampled (that is, the If condition turns out to be false in Line 9).

Now we show the expression (9) still holds for $j = k + 1$. Following the above discussion of the case $j = k$, we have

$$\begin{aligned}\Pr[Z_1 = k + 1] &= \Pr[E_1^{k+1}] + \Pr[E_2^{k+1}] + \cdots + \Pr[E_{k+1}^{k+1}] \\ &= \alpha_1 (1 - p_1)^{k} p_1 \cdot \frac{p_{k+1}}{p_1} + \alpha_2 (1 - p_2)^{k-2} p_2 \cdot \frac{p_{k+1}}{p_2} + \cdots + \alpha_{k+1} p_{k+1} \\ &= \left(\sum_{i=1}^{k} \alpha_i (1 - p_i)^{k+1-i} \right) \cdot p_{k+1} + \alpha_{k+1} \cdot p_{k+1}.\end{aligned}$$

On the other hand, the value α_{k+1} is the probability that SKIP arrives at position k, but fails to sample x_k. We can compute it as follows:

$$\begin{aligned}\alpha_{k+1} &= \sum_{i=1}^{k} \alpha_i (1 - p_i)^{k-i} p_i \cdot (1 - \frac{p_k}{p_i}) \\ &= \sum_{i=1}^{k} \alpha_i (1 - p_i)^{k-i} \cdot (p_i - p_k)\end{aligned}$$

Therefore, plugging the expression of α_{k+1} into $\Pr[Z_1 = k+1]$, we obtain that

$$\begin{aligned}
\Pr[Z_1 = k+1] &= \left(\sum_{i=1}^{k} \alpha_i (1-p_i)^{k+1-i}\right) \cdot p_{k+1} + \left(\sum_{i=1}^{k} \alpha_i (1-p_i)^{k-i} \cdot (p_i - p_k)\right) \cdot p_{k+1} \\
&= \left(\sum_{i=1}^{k} \alpha_i (1-p_i)^{k-i}\right) \cdot (1-p_k) p_{k+1} \\
&= p_{k+1} \prod_{i=1}^{k} (1-p_i).
\end{aligned}$$

By induction, the probability distribution (9) holds for all elements in the input sequence. This completes the poof of the conditional probability distribution (7).

Then we show how to prove the expression (8). In fact, it is rather straightforward if we utilize the above result of the conditional distribution. The event $Z_{m+1} > h$ given $Z_m = j_m$ implies that all the elements $j_m + 1, j_m + 2, \ldots, h$ are not sampled. Thus, we compute the probability as follows:

$$\begin{aligned}
\Pr[Z_{m+1} > h | Z_m = j_m] &= 1 - (\Pr[Z_{m+1} = j_m + 1 | Z_m = j_m] + \cdots + \Pr[Z_{m+1} = j_m + 1 | Z_m = j_m]) \\
&= 1 - \left(p_{m+1} + (1-p_{m+1}) p_{m+2} + \cdots + \prod_{i=j_m+1}^{h-1} (1-p_i) \cdot p_h\right) \\
&= \prod_{i=j_m+1}^{h} (1-p_i).
\end{aligned}$$

Putting together all the above results, we finally complete the proof of Lemma 3, providing a guarantee for the correctness of SKIP.

Time Complexity of SKIP. We will show the optimality of SKIP, in the sense that it achieves the asymptotically tight time complexity in [11].

Theorem 4. *The algorithm SKIP takes an expected time of $O(\mu)$ if $\mu \geq \frac{1}{2}\log h$, and $O\left(1 + \frac{\log h}{\log((\log h)/\mu)}\right)$ otherwise.*

Proof. Here we only give a proof sketch for Lemma 4. A formal proof is included in the appendix. To prove the lemma, we make a claim that SKIP takes no more sampling trials (i.e., the number of geometric distribution samplings) than BUCKET with any fixed β, given a fixed input sequence. If the claim is correct, we immediately obtain the lemma since for any μ, there exists some β such that BUCKET achieves the asymptotically tight time complexity as discussed in Section 4.2.

Fortunately, the claim does hold. For BUCKET, the success probability of the geometric distribution sampling is fixed as p_{β^k} for bucket B_k ($k = 0, 1, 2, \ldots$). In contrast, for SKIP the geometric distribution sampling from position $i \in [\beta^k, \beta^{k+1})$ (i.e., inside bucket B_k) has a success probability $p_i \leq p_{\beta^k}$, as the input sequence is in descending order. Note that the expectation of a geometrically distributed random variable with success probability p is known as $1/p$. Therefore, the expectation of the sampling outcome of SKIP in each bucket is not smaller than that of BUCKET. Besides, for each bucket B_k, BUCKET has to start sampling from the first element (namely β^k), whereas SKIP starts sampling from position $i > \beta^k$, which follows the last sampling trial in bucket B_{k-1}. Based on these two observations, it is straightforward to conclude that SKIP takes no more sampling trials than BUCKET. Thus, we complete the proof of Theorem 4.

Algorithm 7: HIST(G, k, ϵ, δ)

$\epsilon_1 = \epsilon_2 = \epsilon/2$, $\delta_1 = \delta_2 = \delta/2$;
S_b^* = SentinelSet$(G, k, \epsilon_1, \delta_1)$;
S_k^* = IM-Sentinel$(G, k, \epsilon, S_b^*, \epsilon_2, \delta_2)$;
return S_k^*;

Time Complexity under General IC Model. Next we are going to show the time complexity bound of SUBSIM under general IC model with SKIP. The time complexity function $\frac{\log h}{\log((\log h)/\mu)}$ is associated with the value μ, which brings us some difficulties in analyzing computational cost. Here we only consider two common cases: $\mu = O(1)$ and $\mu = \Theta(\log h)$.

Case 1: $\mu = O(1)$. The time complexity of SKIP becomes $O\left(1 + \frac{\log h}{\log\log h}\right)$. Define $\theta(x) = \frac{\log x}{\log\log x}$. To follow a similar derivation of Theorem 1, the concavity of $\theta(x)$ is required. The following lemma is used to prove the concavity of $\theta(x)$.

Lemma 7. *[10] Let f be a concave function and g a non-decreasing and concave function. Assume that f and g are twice-differentiable. Define the function f by $\theta(x) = g(f(x))$ for all x. Then $\theta(x)$ is concave.*

Let $f(x) = \log(x)$ and $g(x) = \frac{x}{\log x}$. Then we have $\theta(x) = g(f(x))$. Obviously f is a concave function for $x > 0$. Besides, since $g'(x) = \ln 2 \cdot \frac{\ln x - 1}{\ln^2 x} \geq 0$ and $g''(x) = \ln 2 \cdot \frac{(2 - \ln x)}{x \ln^3 x} \leq 0$ for $x \geq e^2$, it is obtained that $g(x)$ is non-decreasing and concave. According to Lemma 7, we can draw the conclusion that $\theta(x)$ is concave for $x \geq e^{e^2}$.

Since $\theta(x)$ is concave, then we follow a similar derivation in Section 3.2, and conclude that the time complexity of SUBSIM under general IC model is $O\left(k \cdot \frac{\log(m/n)}{\log\log(m/n)} \cdot n \cdot \log n \cdot \epsilon^{-2}\right)$. It improves existing solution by $O\left(\frac{m}{n} \cdot \frac{\log\log(m/n)}{\log(m/n)}\right)$.

Case 2: $\mu = \Theta(\log h)$. In this setting, the expected cost of SKIP is $O(\log h)$ according to Theorem 4. Therefore, the time complexity of SUBSIM under general IC model becomes $O\left(k \cdot \log(m/n) \cdot n \cdot \log n \cdot \epsilon^{-2}\right)$, improving existing solutions by $O\left(\frac{m}{n} \cdot \frac{1}{\log(m/n)}\right)$.

5 HIGHLY INFLUENTIAL SCENARIOS

In highly influential scenarios, i.e., high influence networks, one of the biggest challenges of existing RR-set-based solutions is that the average size of random RR sets is usually very large, which incurs high running time and memory consumption. Therefore, one natural question is: can we reduce the average size of random RR sets? If the answer is yes, then such a new solution might outperform existing solutions. Motivated by this, we propose *Hit-and-Stop (HIST)* algorithm to overcome the weakness of existing RR-set-based IM algorithms by dramatically decreasing the average size of random RR sets. In particular, a sentinel set S_b^* is selected in the first phase, and with the help of S_b^*, subsequent RR sets can be generated efficiently in the second phase since the generation of an RR set can stop as soon as it reaches any node in S_b^*. We denote this RR set generation algorithm to terminate when it reaches a sentinel set as *RR set-with-Sentinel* algorithm (Algorithm 8).

At a high level, HIST consists of two phases as follows:

- Sentinel Set Selection. This phase seeks for a size-b node set S_b^* that satisfies $\mathsf{I}_C(S_b^*) \geq (1 - (1 - 1/k)^b - \epsilon_1) \cdot \mathsf{I}_C(S_k^o)$ with high probability, where S_k^o is the optimal seed set.

Algorithm 8: RR set-with-Sentinel(G, S_b^*)

1 The steps are similar to that of Algorithm 2 except that it terminates the traversal and returns the RR set when a node $v \in S_b^*$ is activated.

- IM-Sentinel. This phase computes a size-$(k-b)$ seed set S_{k-b}^*, and returns $S_{k-b}^* \cup S_b^*$ as the final result S_k^*.

In the *sentinel set selection* phase, we aim to use only a small number of samples to find a sentinel set S_b^* of b nodes. When $b = k$, the sentinel set selection phase falls into existing IM algorithms that provide a $(1 - (1 - 1/k)^k - \epsilon)$ ($\approx 1 - 1/e - \epsilon$) approximate solution. When $b \ \ddot{Y} \ k$, even though the sentinel set selection phase cannot provide a $1 - 1/e - \epsilon$ approximate solution, it can still provide $1 - (1 - 1/k)^b - \epsilon$ approximate solution (as we will prove in Lemma 8). When b is sufficiently small (much smaller than k), we only need to provide a very loose approximation for set S_b^*, and it allows us to use a much smaller number of random RR sets to find a size-b seed set that provides $1 - (1 - 1/k)^b - \epsilon$ approximate solution compared to solving the IM problem. As we will see, with such a loose approximation on S_b^*, we can still provide approximation guarantee after the second phase, i.e., the *IM-sentinel* phase. To explain, we will compensate the first phase by sampling more random RR sets in the second phase. However, in the second phase, the generation of a random RR set can terminate as soon as any node in sentinel set S_b^* is hit. Therefore, the cost to generate a random RR set can be significantly reduced. Our HIST achieves up to 2 orders of magnitude speedup over existing solutions, which shows the effectiveness of our proposed solution. The pseudocode of the HIST algorithm is shown in Algorithm 7, which is self-explanatory. Notice that we set $\epsilon_1 = \epsilon_2 = \epsilon/2$ so that the final error can be bounded by $1 - 1/e - \epsilon_1 - \epsilon_2 = 1 - 1/e - \epsilon$. Similarly, we set $\delta_1 = \delta_2 = \delta/2$ since both phases have a failure probability of $\delta/2$, and by taking a union bound, the failure probability of the HIST algorithm is $\delta_1 + \delta_2 = \delta$. Next, we present more details of the two phases.

5.1 Sentinel Set Selection Phase

Algorithm 10 shows the pseudocode for the sentinel set selection phase. The main framework is similar to existing IM algorithms in that we sample a certain number of RR sets to see if the approximation ratio is satisfied. If not, we double the number of RR sets and continue the steps until the bound holds. In each iteration, we select nodes with greedy algorithms and choose a sentinel set S_b^* with proper size b.

Node selection with modified greedy. Algorithm 10 Lines 5-15 show the process of finding a sentinel set. If the size b is fixed, we will include the first b nodes selected by the greedy algorithm and make them as the candidate of the sentient set. If this candidate set provides the approximation guarantee (Algorithm 10 Lines 11-12), we return it as the sentinel set.

Recap that the sentinel set we select will be used to facilitate the second phase. In particular, any RR set in the second phase will terminate when it hits a node in the sentinel set. In the standard greedy algorithm, however, it only cares about the *marginal coverage* (Ref. the definition in Section 2.2) in each iteration, and selects the node with the maximum marginal coverage with respect to the set of nodes selected in previous iterations. This does not differentiate two nodes when they share the same maximum marginal coverage but one node has a larger out-degree than the other. However, in our case, the node with a larger out-degree is obviously more preferred since it is more likely to be hit, especially when we only select a sentinel set with a small size. Therefore, we modify the greedy algorithm slightly (Algorithm 9) so as to better achieve the goal. When two

Algorithm 9: Greedy-Degree$(G, \mathcal{R}, k)$

1 The steps are similar to that of Algorithm 1 except Line 3: if multiple nodes have maximum marginal coverage, choose the one with the largest out-degree.

Algorithm 10: SentinelSet$(G, k, \epsilon_1, \delta_1)$

1 Set $\theta_0 = 3 \cdot \ln(1/\delta_1)$ and θ_{max} according to Eqn. 10;
2 Generate random RR sets $\mathcal{R}_1$ with $|\mathcal{R}_1| = \theta_0$;
3 $i_{max} \leftarrow \lceil \log_2 \frac{\theta_{max}}{\theta_0} \rceil$;
4 **for** $i = 1$ *to* i_{max} **do**
5 Generate a size-k seed set S_k^* by invoking Algorithm 9 with $\mathcal{R}_1$ as the input;
6 Estimate the lower bound $\hat{\mathsf{I}}_{\mathsf{C}}^{-}(S_a^*)$ based on the result of Line 5, where $a \in \{1 \dots k\}$;
7 Compute $\mathsf{I}_{\mathsf{C}}^{+}(S_k^o)$ by Eqn. 2, setting $\delta_u = \frac{\delta_1}{3i_{max}}$;
8 Let b be the maximum a such that $\hat{\mathsf{I}}_{\mathsf{C}}^{-}(S_a^*)/\mathsf{I}_{\mathsf{C}}^{+}(S_k^o) \geq (1 - (1 - \frac{1}{k})^a - \epsilon_1)$;
9 Generate a set $\mathcal{R}_2$ of random RR sets with $|\mathcal{R}_2| = |\mathcal{R}_1|$ by invoking RR set-with-Sentinel ;
10 Compute $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)$ by Eqn. 1; set $\delta_l = \frac{\delta_1}{6i_{max}}$;
11 **if** $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)/\mathsf{I}_{\mathsf{C}}^{+}(S_k^o) \geq (1 - (1 - \frac{1}{k})^b - \epsilon_1)$ **then**
12 **return** S_b^*;
13 Increase the size of $\mathcal{R}_2$ to $4|\mathcal{R}_1|$ and compute $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)$ again;
14 **if** $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)/\mathsf{I}_{\mathsf{C}}^{+}(S_k^o) \geq (1 - (1 - \frac{1}{k})^b - \epsilon_1)$ **then**
15 **return** S_b^*;
16 double the size of $\mathcal{R}_1$;
17 **return** S_b^*;

nodes share the same *marginal coverage*, we select the node with a larger out-degree. Notice that this brings at most additional $O(k \cdot n \cdot \log n)$ cost and does not affect the final time complexity of the HIST algorithm. In this case, we are more likely to select influential nodes (that get hit it selects) in Algorithm 9 compared to Algorithm 1 which regards all nodes with the same importance as long as their marginal coverage is the same. In Section 8, we experimentally evaluate the effectiveness of our Greedy-Degree algorithm. It is compared with the standard greedy algorithm and another variant of the greedy algorithm, Greedy-Cost, which attempts to reduce the average size of RR sets from a cost reduction perspective. Please see Appendix B for more details.

Choosing the sentinel set S_b^* with proper size. A naive way to determine the size of the sentinel set is to choose a constant and apply it to all choice of k. However, such a strategy may not make full use of the pruning power of the sentinel set. Therefore, we aim to automate the process of the choice of b. Notice that there is a trade-off between the size b and the speedup of the query efficiency. On the one hand, if b is too small, we have less chance to hit the sentinel set in the second phase, providing inferior speedup. Hence, we hope that the size b to be as large as possible. On the other hand, if b is too large, it is similar to solving the original IM problem. Hence, a small sample size will not help provide the required approximation ratio. To get a good trade-off of these two, i.e., the cost of sampling in the first phase and the benefit we can bring to the second phase, we provide a solution to automatically find the choice of b as large as possible that can satisfy the constraint given the generated RR sets. To explain, given the set $\mathcal{R}_1$ of RR sets, we first apply

Algorithm 9 to select a seed set S_k^*, and we denote S_a^* $(1 \le a \le k)$ as the set of nodes selected by the first a iterations in the modified greedy algorithm. Then, we apply Equation 2 to derive an upper bound $\mathsf{I}_{\mathsf{C}}^{+}(S_k^o)$ for $\mathsf{I}_{\mathsf{C}}(S_k^o)$. However, we can not use $\mathcal{R}_1$ to derive a lower bound of $\mathsf{I}_{\mathsf{C}}(S_a^*)$. To explain, the selected set S_a^* depends on $\mathcal{R}_1$ and we cannot apply the concentration bounds to S_a^*. Therefore, we apply the concentration bound to derive an estimation of the lower bound on $\mathsf{I}_{\mathsf{C}}(S_a^*)$, denoted as $\hat{\mathsf{I}}_{\mathsf{C}}^{-}(S_a^*)$, as if $\mathcal{R}_1$ and S_a^* were independent. Then, we select the maximum a (Algorithm 10 Line 8) such that it satisfies:

$$\hat{\mathsf{I}}_{\mathsf{C}}^{-}(S_a^*)/\mathsf{I}_{\mathsf{C}}^{+}(S_k^o) \ge (1-(1-\frac{1}{k})^a - \epsilon_1),$$

and set b to this maximum a. However, since this is only an estimation of the lower bound, we generate another set $\mathcal{R}_2$ of RR set and $\mathcal{R}_2$ is independent of S_b^*. Then, we can apply concentration bound to derive the lower bound $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)$ using Equation 1. Given $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)$, we are able to check if S_b^* satisfies the approximation ratio (Algorithm 10 Line 11), i.e.,

$$\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)/\mathsf{I}_{\mathsf{C}}^{+}(S_k^o) \ge (1-(1-\frac{1}{k})^a - \epsilon_1).$$

If the approximation ratio is not satisfied, with the existing paradigm, we will simply double the size of $\mathcal{R}_1$ and repeat above process. However, since now we are only to estimate the influence of S_b^*, we can stop when any node in this set is hit. Hence, the *RR set-with-Sentinel* algorithm can be applied here and tends to save much time. To take this advantage, if we find that S_b^* violates the approximation guarantee, we first increase the size of $\mathcal{R}_2$ and try to provide a tighter lower bound $\mathsf{I}_{\mathsf{C}}^{-}(S_b^*)$ for S_b^* (Algorithm 10 Lines 13-15). We increase the size of $\mathcal{R}_2$ until $|\mathcal{R}_2| = 4 \cdot |\mathcal{R}_1|$ and stop increasing afterwards since it actually indicates that S_b^* we selected is most likely not good enough to provide the approximation ratio. Therefore, we select another set S_b^* by doubling the size of $\mathcal{R}_1$ (Algorithm 10 Line 16). We repeat the whole process until we find the seed set S_b^* satisfying the required approximation ratio.

Stopping condition. We now give the following lemma to establish the stopping condition of Algorithm 10. It provides a bound on the number of random RR sets required in $\mathcal{R}_1$ in the sentinel set selection phase.

Lemma 8. *Let $\mathcal{R}_1$ be the set of random RR sets generated by Algorithm 10 and S_b^* be a size-b node set selected by Algorithm 9 on $\mathcal{R}_1$. Given ϵ' and δ', if the size of $\mathcal{R}_1$ satisfies*

$$|\mathcal{R}_1| \ge \frac{2n\left(\sqrt{(1-x^b)\ln\frac{2}{\delta'}} + \sqrt{(1-x^b)(\ln\binom{n}{b} + \ln\frac{2}{\delta'})}\right)^2}{\epsilon'^2 \cdot \mathsf{I}_{\mathsf{C}}(S_k^o)},$$

where $x = 1-\frac{1}{k}$, then with at least $1-\delta'$ probability,

$$\mathsf{I}_{\mathsf{C}}(S_b^*) \ge (1-(1-1/k)^b - \epsilon')\mathsf{I}_{\mathsf{C}}(S_k^o).$$

Further notice that the size of $\mathcal{R}_2$ solely depends on $\mathcal{R}_1$, and is only constant times the size of $\mathcal{R}_1$, and therefore we omit its discussion. According to Lemma 8, by replacing $\mathsf{I}_{\mathsf{C}}(S_k^o)$ with k, $\ln\binom{n}{b}$ with $\ln\binom{n}{k}$, $1-x^b$ with 1, and setting $\epsilon' = \epsilon_1, \delta' = \delta_1/3$, we define the maximum number of random RR sets θ_{max} as follows:

$$\theta_{max} = \frac{2n\left(\sqrt{\ln\frac{6}{\delta_1}} + \sqrt{(\ln\binom{n}{k} + \ln\frac{6}{\delta_1})}\right)^2}{\epsilon_1^2 \cdot k}. \tag{10}$$

That is, if the size of the set $\mathcal{R}_1$ is θ_{max}, the seed set S_b^* selected by Algorithm 9 guarantees $(1-(1-1/k)^b-\epsilon_1)$ approximation of $\mathsf{I}_\mathsf{C}(S_k^o)$ with at least $1-\delta_1/3$ probability. The reason of choosing the probability of $1-\delta_1/3$, rather than $1-\delta_1$, will be explained shortly.

In terms of the initial setting, for a random variable in the range of $[0,1]$ with an expectation to be μ, the Monte-Carlo method requires at least $3\ln(1/\delta)/\mu/\epsilon^2$ [19] so as to provide an estimation of μ with ϵ-relative error guarantee. Hence, we set the initial number θ_0 to be $3\ln(1/\delta_1)$ (Algorithm 10 Line 1), which is the case when the random variable has an expectation of 1 and the relative error is close to 1.

Failure probability. Here we explain why Algorithm 10 ensures $(1-(1-1/k)^b-\epsilon_1)$ approximation with at least $1-\delta_1$ probability. The algorithm has at most i_{max} iterations. In the last iteration, no matter whether $\mathsf{I}_\mathsf{C}^-(S_b^*)/\mathsf{I}_\mathsf{C}^+(S_k^o)$ reaches the approximation threshold or not, it returns S_b^* as the final seed set. As shown in Lemma 8, θ_{max} RR samples ensure that the failure probability of S_b^* being unqualified, i.e. $\mathsf{I}_\mathsf{C}(S_b^*)$ $\ddot{Y}$ $(1-(1-1/k)^b-\epsilon_1)\mathsf{I}_\mathsf{C}(S_k^o)$, is less than $\delta_1/3$. In each of the first $i_{max}-1$ iterations, by the union bound, the failure probability that the algorithm terminates with an unqualified set S_b^* is at most $\frac{\delta_1}{3i_{max}}+2\cdot\frac{\delta_1}{6i_{max}}=\frac{2\delta_1}{3i_{max}}$ (the lower bound is computed twice at most). The total failure probability of the first $i_{max}-1$ iterations is at most $2\delta_1/3$. Therefore, the failure probability of Algorithm 10 is at most δ_1.

5.2 IM-Sentinel Phase

Algorithm 11 shows the pseudocode of the IM-Sentinel phase. In this phase, we apply Algorithm 8 to sample random RR sets and immediately terminate when the RR set reaches a node in S_b^*. For the remaining parts, they are similar to that of the first phase. In particular, Algorithm 11 initializes the sample size of the RR sets to be $3\ln(1/\delta_2)$ and set the maximum number of RR set according to Equation 11 (Algorithm 11 Line 1). Then, in each iteration, it samples a set $\mathcal{R}_1$ and a set $\mathcal{R}_2$ of random RR sets with equal size. It uses $\mathcal{R}_1$ to find the seed set S_k^* by invoking Algorithm 9 and derives the upper bound $\mathsf{I}_\mathsf{C}^+(S_k^o)$ (Algorithm 11 Lines 5-8), and uses the other set $\mathcal{R}_2$ to derive the lower bound $\mathsf{I}_\mathsf{C}^-(S_k^*)$ (Algorithm 11 Line 9). When the approximation ratio is satisfied, we return the seed set S_k^* (Algorithm 11 Line 10-11). Otherwise, we double the size of $\mathcal{R}_1$ and $\mathcal{R}_2$ and then repeat the above process.

The main difference is that, when generating $\mathcal{R}_1$ and $\mathcal{R}_2$, we can apply Algorithm 8 to effectively reduce the size of a random RR set. Besides, when we feed $\mathcal{R}_1$ to Algorithm 9 to greedily select the remaining $k-b$ nodes, we remove the RR sets that hit any node in S_b^* since such RR sets will bring zero marginal coverage to other nodes (Algorithm 11 Line 5).

Stopping condition. Here we provide another lemma to bound the size of $\mathcal{R}_1$ in Algorithm 11.

Lemma 9. *Let S_b^* be the seed set returned by Algorithm 10. Given ϵ' and δ', if the size of $\mathcal{R}_1$ satisfies*

$$|\mathcal{R}_1| \geq \frac{2n\cdot\left(\sqrt{\ln\frac{3}{\delta'}}+\sqrt{(1-1/e)(\ln\binom{n-b}{k-b}+\ln\frac{3}{\delta'})}\right)^2}{\mathsf{I}_\mathsf{C}(S_k^o)\epsilon'^2},$$

then with at least $1-\delta'$ probability, the selected S_{k-b}^* satisfies

$$\mathsf{I}_\mathsf{C}(S_b^*\cup S_{k-b}^*)\geq(1-1/e-\epsilon_1-\epsilon')\mathsf{I}_\mathsf{C}(S_k^o).$$

Algorithm 11: IM-Sentinel$(G, k, \epsilon, S_b^*, \epsilon_2, \delta_2)$

1 Set $\theta_0 = 3 \cdot \ln(1/\delta_2)$ and θ_{max} according to Eqn. 11;
2 Generate $\mathcal{R}_1$ and $\mathcal{R}_2$ with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_0$ by utilizing RR set-with-Sentinel;
3 $i_{max} \leftarrow \lceil \log_2 \frac{\theta_{max}}{\theta_0} \rceil$;
4 **for** $i = 1$ *to* i_{max} **do**
5 $\quad \mathcal{R}_1' \leftarrow \{R : R \in \mathcal{R}_1, R \cap S_b^* = \emptyset\}$;
6 $\quad$ Select a size-$(k-b)$ seed set S_{k-b}^* by invoking Algorithm 9 on $\mathcal{R}_1'$;
7 $\quad S_k^* \leftarrow S_b^* \cup S_{k-b}^*$;
8 $\quad$ Compute $\mathsf{I}_{\mathsf{C}}^+(S_k^o)$ by Eqn. 2 with $\mathcal{R}_1$; set $\delta_u = \frac{\delta_2}{3i_{max}}$;
9 $\quad$ Compute $\mathsf{I}_{\mathsf{C}}^-(S_k^*)$ by Eqn. 1 with $\mathcal{R}_2$; set $\delta_l = \frac{\delta_2}{3i_{max}}$;
10 $\quad$ **if** $\mathsf{I}_{\mathsf{C}}^-(S_k^*)/\mathsf{I}_{\mathsf{C}}^+(S_k^o) \geq (1 - 1/e - \epsilon)$ **then**
11 $\quad\quad$ **return** S_k^*;
12 $\quad$ double the size of $\mathcal{R}_1$ and $\mathcal{R}_2$ by utilizing RR set-with-Sentinel;
13 **return** S_k^*;

According to Lemma 9, we replace $\mathsf{I}_{\mathsf{C}}(S_k^o)$ with k, set $\delta' = \delta_2/3, \epsilon' = \epsilon_2$, and define the maximum number of RR sets in the IM-Sentinel phase as

$$\theta_{max} = \frac{2n \cdot \left(\sqrt{\ln \frac{9}{\delta_2}} + \sqrt{(1-1/e)(\ln \binom{n-b}{k-b} + \ln \frac{9}{\delta_2})}\right)^2}{\epsilon_2^2 \cdot k}. \tag{11}$$

That is, if the size of $\mathcal{R}_1$ is θ_{max}, the seed set S_{k-b}^* obtained in Algorithm 11 Line 6 guarantees $\mathsf{I}_{\mathsf{C}}(S_b^* \cup S_{k-b}^*) \geq (1 - 1/e - \epsilon_1 - \epsilon_2)\mathsf{I}_{\mathsf{C}}(S_k^o)$ with at least $1 - \delta_2/3$ probability.

Failure probability. Like the analysis of Algorithm 10, the total failure probability of the first $i_{max} - 1$ iterations is $2\delta_2/3$. Taking the failure probability of $\delta_2/3$ in the last iteration into consideration, the failure probability of Algorithm 11 is δ_2.

Remark. By using multiple (more than two) phases, it is likely to have a smaller average size of RR sets in the later phases since there exists a larger sentinel set. However, there is no free lunch by bringing more phases. With multiple phases, it will introduce the following costs:

- If HIST consists of l phases, the error threshold of each phase will be ϵ/l. It implies that we have to compute a more accurate result for each phase.
- The RR sets in the former phases cannot be reused in the subsequent phases, which may cause heavy computational overhead.

Thus for ease of analysis, we take the simplest case in our HIST solution.

5.3 Theoretical Analysis

In this section, we provide the proofs of Lemma 8 and 9.

Proof of Lemma 8. We first give three lemmas that will be used in the proof of Lemma 8.

Lemma 10 ([32]). *Let S_b^* be the seed set selected by Algorithm 9. Let $x = (1 - 1/k)$, then $\Lambda_{\mathcal{R}}(S_b^*) \geq 1 - x^b \quad \Lambda_{\mathcal{R}}(S_k^o)$.*

Denote the size of $\mathcal{R}$ as θ. Since $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o)$ is an unbiased estimator of $\mathsf{I}_{\mathsf{C}}(S_k^o)$. If θ is large enough, $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o)$ should be close to $\mathsf{I}_{\mathsf{C}}(S_k^o)$, as shown in the following lemma.

Lemma 11 ([47]). *Given δ_1', ϵ_1', and $\theta_1 = \frac{2n\cdot\ln(1/\delta_1')}{\mathsf{I}_C(S_k^o)\cdot\epsilon_1'^2}$, if $\theta \geq \theta_1$, $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o) \geq (1-\epsilon_1')\mathsf{I}_C(S_k^o)$ holds with $1-\delta_1'$ probability.*

The proof of Lemma 11 is presented in [47] (Lemma 3). Here, if θ is large, $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*)$ is close to $\mathsf{I}_C(S_b^*)$. Based on Lemmas 10-11, we have:

$$\frac{n}{\theta}\cdot\Lambda_{\mathcal{R}}(S_b^*) \geq \quad 1-x^b \quad (1-\epsilon_1')\mathsf{I}_C(S_k^o). \tag{12}$$

Hence, it is possible for us to build a connection between $\mathsf{I}_C(S_b^*)$ and $\mathsf{I}_C(S_k^o)$.

Lemma 12. *Given δ_2', ϵ_1' $\ddot{Y}$ ϵ', if Equation 12 holds and*

$$\theta \ \text{¡} \ \theta_2 = \frac{2(1-x^b)\cdot n\cdot \ \ln \ \binom{n}{b} \ + \ln\frac{1}{\delta_2'}}{\mathsf{I}_C(S_k^o)\cdot \ \epsilon' - (1-x^b)\cdot\epsilon_1'^{\ 2}},$$

then with at least $1-\delta_2'$ probability, we have $\mathsf{I}_C(S_b^*) \geq (1-x^b-\epsilon')\mathsf{I}_C(S_k^o)$.

Proof. The proof follows similar steps as Lemma 4 in [47]. Let S_b be an arbitrary size-b node set selected from V. We say S_b is bad if $\mathsf{I}_C(S_b)$ $\ddot{Y}$ $(1-x^b-\epsilon')\mathsf{I}_C(S_k^o)$. Define a random variable x_i for each $R_i \in \mathcal{R}$, such that $x_i = 1$ if $R_i \cap S_b \prec \emptyset$, and $x_i = 0$ if otherwise. Define $p = \mathsf{I}_C(S_b)/n$. By definition, we have p $\ddot{Y}$ $(1-x^b-\epsilon')\mathsf{I}_C(S_k^o)/n$. Let $\epsilon_2' = \epsilon' - (1-x^b)\cdot\epsilon_1'$.

$$\Pr\left[\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b) - \mathsf{I}_C(S_b) \geq \epsilon_2'\mathsf{I}_C(S_k^o)\right] = \Pr\left[\sum_{i=1}^{\theta} x_i - \theta p \geq \frac{\epsilon_2'\mathsf{I}_C(S_k^o)}{np}\cdot\theta p\right]$$

$$\leq \exp\left(-\frac{\epsilon_2'^2\mathsf{I}_C(S_k^o)^2}{2n^2p + \frac{2}{3}\epsilon_2' n\mathsf{I}_C(S_k^o)}\cdot\theta\right) \leq \exp\left(-\frac{\epsilon_2'^2\mathsf{I}_C(S_k^o)}{2n(1-x^b-\epsilon') + \frac{2}{3}\epsilon_2' n}\cdot\theta\right)$$

$$\leq \exp\left(-\frac{\epsilon_2'^2\mathsf{I}_C(S_k^o)}{2n(1-x^b)}\cdot\theta_2\right) \leq \delta_2'/\binom{n}{b}.$$

Hence, if θ ¡ θ_2, the probability of bad S_b^* is at most δ_2', since there exists only $\binom{n}{b}$ size-b node sets and each has at most $\delta_2'/\binom{n}{b}$. That is, with at least $1-\delta_2'$ probability,

$$\mathsf{I}_C(S_b^*) \geq \frac{n}{\theta}\cdot\Lambda_{\mathcal{R}}(S_b^*) - \epsilon_2'\mathsf{I}_C(S_k^o) \geq (1-x^b-\epsilon')\mathsf{I}_C(S_k^o).$$

Hence, the lemma follows.

Now we give the proof of Lemma 8. Based on Lemma 11 and Lemma 12 and by the union bound, if θ ¡ $\max(\theta_1,\theta_2)$, it holds that $\mathsf{I}_C(S_b^*) \geq (1-x^b-\epsilon')\mathsf{I}_C(S_k^o)$ with at least $1-\delta_1'-\delta_2'$ probability. Set $\delta_1' = \delta_2' = \delta'/2$ and $\theta_1 = \theta_2$, denoted as θ', we have

$$\theta' = \frac{2n\ \left(\sqrt{(1-x^b)\ln\frac{2}{\delta'}} + \sqrt{(1-x^b)(\ln\binom{n}{b} + \ln\frac{2}{\delta'})}\right)^2}{\epsilon'^2\cdot\mathsf{I}_C(S_k^o)}.$$

Hence, if θ ¡ θ', S_b^* satisfies $\mathsf{I}_C(S_b^*) \geq (1-x^b-\epsilon')\mathsf{I}_C(S_k^o)$ with at least $1-\delta'$ probability.

Proof of Lemma 9. We first give several lemmas that will be used in the proof of Lemma 9.

Lemma 13. *Let S_b^* be the seed set returned by Algorithm 10. Let S_{k-b}^* be the seed set generated in Algorithm 11 Line 6 on a set $\mathcal{R}$ of random RR sets. Then we have $\Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}^*) \geq (1-x^{k-b})\Lambda_{\mathcal{R}}(S_k^o) + x^{k-b}\Lambda_{\mathcal{R}}(S_b^*)$, where $x = 1-1/k$.*

Proof. Let $S_j^*(1 \le j \le k-b)$ be the set of nodes selected in the first j iterations of the generation of S_{k-b}^*, and $M_j(1 \le j \le k-b)$ be a union of S_b^* and S_j^*, i.e. $M_j = S_b^* \cup S_j^*$. By the submodularity property of coverage function $\Lambda_{\mathcal{R}}(\cdot)$,

$$\Lambda_{\mathcal{R}}(S_k^o) \le \Lambda_{\mathcal{R}}(S_k^o \cup M_j) \le \Lambda_{\mathcal{R}}(M_j) + \sum_{v \in S_k^o \setminus M_j} \Lambda_{\mathcal{R}}(v|M_j)$$
$$\le \Lambda_{\mathcal{R}}(M_j) + k \left(\Lambda_{\mathcal{R}}(M_{j+1}) - \Lambda_{\mathcal{R}}(M_j)\right).$$

Let $\gamma_j = \Lambda_{\mathcal{R}}(S_k^o) - \Lambda_{\mathcal{R}}(M_j)$. Then we have: $\gamma_{j+1} \le (1 - \frac{1}{k})\gamma_j = x\gamma_j$. Recursively, we have $\gamma_{k-b} \le x^{k-b}\gamma_0$. By the definition of γ_j and M_j, we derive that:

$$\gamma_0 = \Lambda_{\mathcal{R}}(S_k^o) - \Lambda_{\mathcal{R}}(S_b^*), \quad \gamma_{k-b} = \Lambda_{\mathcal{R}}(S_k^o) - \Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}^*).$$

Hence, it can be obtained that,

$$\Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}^*) \ge (1 - x^{k-b})\Lambda_{\mathcal{R}}(S_k^o) + x^{k-b}\Lambda_{\mathcal{R}}(S_b^*).$$

The lemma is proved.

According to Lemma 11, given δ_1', ϵ_1', and $\theta_1 = \frac{2n \ln \frac{1}{\delta_1'}}{\mathsf{I}_C(S_k^o) \cdot \epsilon_1'^2}$, if the size of $\mathcal{R}$, denoted as θ, is larger than θ_1, it follows that $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o) \ge (1 - \epsilon_1')\mathsf{I}_C(S_k^o)$ with at least $1 - \delta_1'$ probability. In fact, at this moment ($\theta \ge \theta_1$), $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*)$ is close to $\mathsf{I}_C(S_b^*)$. That is the following lemma.

Lemma 14. *Given ϵ_1' and δ_1', if $\theta \ge \theta_1$, it holds that $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*) \ge \mathsf{I}_C(S_b^*) - \epsilon_1'\mathsf{I}_C(S_k^o)$ with at least $1 - \delta_1'$ probability.*

Proof. Define a random variable x_i for each $R_i \in \mathcal{R}$, such that $x_i = 1$ if $S_b^* \cap R_i \neq \emptyset$, and $x_i = 0$ if otherwise. Define $p = \mathsf{I}_C(S_b^*)/n$. Obviously, $\mathsf{I}_C(S_k^o) \ge \mathsf{I}_C(S_b^*) = np$. We have

$$\Pr\left[\frac{n}{\theta} \cdot \Lambda_{\mathcal{R}}(S_b^*) - \mathsf{I}_C(S_b^*) \le -\epsilon_1'\mathsf{I}_C(S_k^o)\right] = \Pr\left[\sum_{i=1}^{\theta} x_i - \theta p \le -\frac{\epsilon_1'\mathsf{I}_C(S_k^o)}{np}\theta p\right]$$
$$\le \exp\left(-\frac{\epsilon_1'\mathsf{I}_C(S_k^o)}{np}^2 \frac{p\theta_1}{2}\right)$$
$$\le \exp\left(-\frac{\epsilon_1'\mathsf{I}_C(S_k^o)}{np}^2 \frac{p\theta_1}{2}\right) \le \delta_1'.$$

The lemma follows.

Combining Lemma 13 and 14, we have that:

Lemma 15. *Given δ_1' and ϵ_1', if $\theta \ge \theta_1$ and $\mathsf{I}_C(S_b^*) \ge (1 - x^b - \epsilon_1)\mathsf{I}_C(S_k^o)$, then with at least $1 - 2\delta_1'$ probability, we have*

$$\frac{n}{\theta} \cdot \Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}^*) \ge (1 - 1/e - \epsilon_1 - \epsilon_1')\mathsf{I}_C(S_k^o). \tag{13}$$

Proof. Based on Lemma 13,

$$\frac{n}{\theta} \cdot \Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}^*) \ge (1 - x^{k-b})\Lambda_{\mathcal{R}}(S_k^o) + x^{k-b}\Lambda_{\mathcal{R}}(S_b^*)$$
$$\ge (1 - x^{k-b})(1 - \epsilon_1')\mathsf{I}_C(S_k^o) + x^{k-b}\left((1 - x^b - \epsilon_1)\mathsf{I}_C(S_k^o) - \epsilon_1'\mathsf{I}_C(S_k^o)\right)$$
$$= (1 - x^k - \epsilon_1' - x^{k-b}\epsilon_1)\mathsf{I}_C(S_k^o) \ge (1 - x^k - \epsilon_1 - \epsilon_1')\mathsf{I}_C(S_k^o).$$

When $\theta \ge \theta_1$, both $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o) \ge (1 - \epsilon_1')\mathsf{I}_C(S_k^o)$ and $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*) \ge \mathsf{I}_C(S_b^*) - \epsilon_1'\mathsf{I}_C(S_k^o)$ hold with at least $1 - \delta_1'$ probability. By union bound, the failure probability is $2\delta_1'$. The lemma follows.

Let ϵ' be the error threshold in the IM-sentinel phase.

Lemma 16. *Given δ_2', ϵ_1' $\ddot{Y}$ ϵ', and*

$$\theta_2 = \frac{2(1-1/e)\cdot n \ \ln \binom{n-b}{k-b} + \ln \frac{1}{\delta_2'}}{\mathsf{I}_C(S_k^o)(\epsilon' - \epsilon_1')^2},$$

if Equation 13 holds and θ ¡ θ_2, then

$$\mathsf{I}_C(S_b^* \cup S_{k-b}^*) \geq (1 - 1/e - \epsilon_1 - \epsilon')\mathsf{I}_C(S_k^o).$$

Proof. Let S_{k-b} be an arbitrary size-$(k-b)$ node set selected from $V \backslash S_b^*$. We say that S_{k-b} is bad, if $\mathsf{I}_C(S_b^* \cup S_{k-b})$ $\ddot{Y}$ $(1-1/e-\epsilon_1-\epsilon')\mathsf{I}_C(S_k^o)$. Then, we can follow the proof steps in Lemma 12 to prove the lemma. Define a random variable x_i for each $R_i \in \mathcal{R}$, such that $x_i = 1$ if $R_i \cap \{S_b^* \cup S_{k-b}\} \prec \emptyset$, and $x_i = 0$ if otherwise. Define $p = \mathsf{I}_C(S_b^* \cup S_{k-b})/n$. By definition, we have p $\ddot{Y}$ $(1-1/e-\epsilon_1-\epsilon')\mathsf{I}_C(S_k^o)/n$. Let $\epsilon_2' = \epsilon' - \epsilon_1'$. Following the proof steps in Lemma 12, we can prove that

$$\begin{aligned}
&\Pr\left[\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}) - \mathsf{I}_C(S_b^* \cup S_{k-b}) \geq \epsilon_2'\mathsf{I}_C(S_k^o)\right] \\
&= \Pr\left[\sum_i x_i - \theta p \geq \frac{\epsilon_2'\mathsf{I}_C(S_k^o)}{np}\cdot\theta p\right] \leq \exp\left[-\frac{\epsilon_2'^2\mathsf{I}_C(S_k^o)^2}{2n^2 p + \frac{2}{3}\epsilon_2' n \mathsf{I}_C(S_k^o)}\cdot\theta\right] \\
&\leq \exp\left[-\frac{\epsilon_2'^2\mathsf{I}_C(S_k^o)}{2n(1-1/e-\epsilon_1-\epsilon') + \frac{2}{3}\epsilon_2' n}\cdot\theta\right] \\
&\leq \exp\left[-\frac{\epsilon_2'^2\mathsf{I}_C(S_k^o)}{2n(1-1/e)}\cdot\theta_2\right] \leq \delta_2' / \binom{n-b}{k-b}.
\end{aligned}$$

Hence, if θ ¡ θ_2, the probability of bad S_{k-b}^* is at most δ_2', since there exist only $\binom{n-b}{k-b}$ size-$(k-b)$ node sets and each has at most $\delta_2' / \binom{n-b}{k-b}$ probability. That is, with at least $1-\delta_2'$,

$$\begin{aligned}
\mathsf{I}_C(S_b^* \cup S_{k-b}^*) &\geq \frac{n}{\theta}\cdot\Lambda_{\mathcal{R}}(S_b^* \cup S_{k-b}^*) - \epsilon_2'\mathsf{I}_C(S_k^o) \\
&\geq (1 - 1/e - \epsilon_1 - \epsilon')\mathsf{I}_C(S_k^o).
\end{aligned}$$

Hence, the lemma follows.

Now we prove Lemma 9. Lemmas 15 and 16 hold with $1-2\delta_1'$ and $1-\delta_2'$ probability, respectively. By union bound, if θ ¡ $\max(\theta_1, \theta_2)$, with $1-2\delta_1'-\delta_2'$ probability, we have that:

$$\mathsf{I}_C(S_b^* \cup S_{k-b}^*) \geq (1 - 1/e - \epsilon_1 - \epsilon')\mathsf{I}_C(S_k^o).$$

By setting $\delta_1' = \delta_2' = \delta'/3$, $\theta_1 = \theta_2$, denoted as θ', we have:

$$\theta' = \frac{2n\cdot\left(\sqrt{\ln\frac{3}{\delta'}} + \sqrt{(1-1/e)(\ln\binom{n-b}{k-b} + \ln\frac{3}{\delta'})}\right)^2}{\mathsf{I}_C(S_k^o)\epsilon'^2}.$$

The lemma follows.

6 FORWARD PROPAGATION

In the previous sections, we propose better solutions to find out a seed set for the IM problem by means of the RR-set-based method. However, if we also hope to get fine-grained propagating information, the RR-set-based approach may not be a good solution. For example, if we have obtained a seed set S, now we hope to estimate the probability that node v is influenced with respect to S for all $v \in G$. One possible solution is sampling sufficient RR sets from each node in the graph G, which incurs prohibitive computations. On the other hand, we might conduct many forward propagations, then the fraction of the simulations in which node v is finally activated could works as an unbiased estimator for the probability that seed set S influences node v. With the subset sampling techniques developed in Section 4, the forward propagation approach might be efficient, compared with the RR-set-based method. Motivated by this, we investigate this approach and provide some new insights.

This section is organized as follows: in Section 6.1, we develop the time cost of conducting a random forward propagation with subset sampling algorithms ISS and SKIP; in Section 6.2, we propose a heuristic condition to indicate when the forward propagation should be chosen in terms of estimating the expected influence of a given seed set.

6.1 Time Complexity

We provide a theorem to bound the expected cost of conducting a random forward propagation. This theorem holds for *both directed and undirected graphs*.

Theorem 5. *Given a graph $G = (V, E)$, a seed set S , if it holds that $\prod_{u \in IN(v)} p(u, v) \leq c$ for any node $v \in V$ where c is a constant number, then the expected cost to conduct a forward propagation is $O(\mathsf{I}_{\mathsf{C}}(S))$ with ISS algorithm, and $O(\log(d_{max}) \cdot \mathsf{I}_{\mathsf{C}}(S))$ with SKIP algorithm where d_{max} is the maximum out-degree of the graph G.*

Proof. Let $p(u, v)$ denote the propagating probability of the edge from node u to node v. Let $\Pr[S \rightarrow u]$ denote the probability that the seed set S activates node u. Let $IN(u)$ be the set of u's in-neighbors and $OUT(u)$ be the set of u's out-neighbors.

Let $E_C(S)$ be the set of edges being alive under the cascade process C from S. That is, given a realization ϕ of the probabilistic graph G by the live/blocked edge approach, for the correspoinding cascade C from S on ϕ, only the outgoing edges in ϕ from the activated nodes by S are included in $E_C(S)$, while other edges in ϕ, though being alive, are excluded. Let $N_C(S)$ be the set of activated nodes by S under the cascade process C. Then the expected cost $E_F(S)$ of running a forward propagation with ISS is given as follows:

$$E_F(S) = \mathsf{I}_{\mathsf{C}}(S) + \mathsf{E}_{C \in \mathsf{C}}\left[|E_C(S)|\right].$$

The above equality is due to the fact that the ISS algorithm take $O(1)$ to sample a live edge, and we have to take at least $O(1)$ for each activated node. Therefore, we charge $O(1)$ cost if an edge is alive and charge no cost if the edge is not alive in the following proof.

We rewrite the expectation $\mathsf{E}_{C \in \mathsf{C}}\left[|E_C(S)|\right]$ as follows:

$$\mathsf{E}_{C \in \mathsf{C}}\left[|E_C(S)|\right] = \mathsf{E}_{C \in \mathsf{C}}\left[\sum_{v \in V} \sum_{u \in IN(v)} \mathbf{1}_{E_C(S)}(u, v)\right] = \sum_{v \in V} \mathsf{E}_{C \in \mathsf{C}}\left[\sum_{u \in IN(v)} \mathbf{1}_{E_C(S)}(u, v)\right]$$

where $\mathbf{1}_{E_C(S)}(u, v)$ is an indicator function such that $\mathbf{1}_{E_C(S)}(u, v) = 1$ if edge $(u, v) \in E_C(S)$ and $\mathbf{1}_{E_C(S)}(u, v) = 0$ otherwise.

Now for node $v \in V$ with $d_{in}(v)$ in-neighbors, we fix an order of these in-neighbors, and then we partition C into $d_{in}(v) + 1$ disjoint sets C_i^v for $i = 0, 1, \ldots, d_{in}(v)$ such that C_i^v is the set of cascades

where v's i-th incoming edge is the first (i.e., smallest-indexed) edge being visited by the cascade for $i \geq 1$, and $\mathbb{C}_0^v$ is the set of cascades where none of the incoming edges is visited (i.e., v is not activated under the cascade). Let u_i be the i-th incoming neighbor of v. Then, for each $v \in V$,

$$\mathbb{E}_{C\in\mathbb{C}}\left[\sum_{u\in IN(v)} 1_{E_C(S)}(u,v)\right] = \sum_{i=1}^{d_{in}(v)} \mathbb{E}_{C\in\mathbb{C}_i^v}\left[\sum_{u\in IN(v)} 1_{E_C(S)}(u,v)\right] \cdot \Pr\left[C \in \mathbb{C}_i^v\right]$$
$$= \sum_{i=1}^{d_{in}(v)} \mathbb{E}_{C\in\mathbb{C}_i^v}\left[1 + \sum_{u\in IN(v)\setminus\{u_i\}} 1_{E_C(S)}(u,v)\right] \cdot \Pr\left[C \in \mathbb{C}_i^v\right]. \quad (14)$$

Now we analyze the term $\mathbb{E}_{C\in\mathbb{C}_i^v}\left[1_{E_C(S)}(u,v)\right]$. Note that for cascade $C \in \mathbb{C}_i^v$,

- For $t = 1, 2, \ldots, i-1$, edge (u_t, v) is not alive by definition, and thus $\sum_{t=1}^{i-1} 1_{E_C(S)}(u_t, v) = 0$;
- For $t = i+1, \ldots, d_{in}(v)$, if u_t is not activated, we have $1_{E_C(S)}(u_t, v) = 0$;
- For $t = i+1, \ldots, d_{in}(v)$, if u_t is activated, we have $1_{E_C(S)}(u_t, v) = 1$ with probability $p(u_t, v)$, and $1_{E_C(S)}(u_t, v) = 0$ with probability $1 - p(u_t, v)$, due to independent propagation.

Therefore, we can obtain that

$$\mathbb{E}_{C\in\mathbb{C}_i^v}\left[\sum_{u\in IN(v)\setminus\{u_i\}} 1_{E_C(S)}(u,v)\right] = \mathbb{E}_{C\in\mathbb{C}_i^v}\left[\sum_{t=i+1}^{d_{in}(v)} 1_{E_C(S)}(u_t,v)\right]$$
$$\leq \sum_{t=i+1}^{d_{in}(v)} p(u_t,v) \leq \sum_{u\in IN(v)\setminus\{u_i\}} p(u,v).$$

Therefore, from (14) and the constraint $\sum_{u\in IN(v)} p(u,v) \leq c$, we have

$$\mathbb{E}_{C\in\mathbb{C}}\left[\sum_{u\in IN(v)} 1_{E_C(S)}(u,v)\right] \leq \sum_{i=1}^{d_{in}(v)} \left(1 + \sum_{u\in IN(v)\setminus\{u_i\}} p(u,v)\right) \cdot \Pr\left[C \in \mathbb{C}_i^v\right]$$
$$\leq (1+c) \sum_{i=1}^{d_{in}(v)} \Pr\left[C \in \mathbb{C}_i^v\right] = (1+c)\Pr[S \to v].$$

Finally, putting them together yields

$$E_F(S) \leq \mathsf{I}_{\mathbb{C}}(S) + (1+c) \sum_{v\in V} \Pr[S \to v] = (2+c)\mathsf{I}_{\mathbb{C}}(S).$$

It completes the proof of the conclusion that the time complexity of a forward-propagation simulation with ISS is bounded by $O(\mathsf{I}_{\mathbb{C}}(S))$.

Let $E'_F(S)$ be the expected cost of conducting a forward propagation with SKIP. According to Theorem 4, we know that SKIP takes a cost not more than $1 + \beta\mu + \log_\beta h$ for any $\beta \geq 2$. Let $\beta = 2$,

and then, we have

$$E'_F(S) = \sum_{u\in V} \Pr[S \to u]\left(1 + 2\sum_{v\in OUT(u)} p(u,v) + \log(d_{out}(u))\right)$$
$$= 2\sum_{u\in V} \Pr[S \to u]\left(1 + \sum_{v\in OUT(u)} p(u,v)\right) + \sum_{u\in V} (\log(d_{out}(u)) - 1)\Pr[S \to u]$$
$$= 2E_F(S) + \sum_{u\in V} (\log(d_{out}(u)) - 1)\Pr[S \to u]$$

where $E_F(S) = \sum_{u\in V} \Pr[S \to u]\left(1 + \sum_{v\in OUT(u)} p(u,v)\right)$ since ISS algorithm samples a live edge with $O(1)$ cost. Note that $E_F(S) \le (c+2)\mathsf{I}_{\mathsf{C}}(S)$ and $\log(d_{out}(u)) \le \log(d_{max})$ for any $u \in V$. Then,

$$E'_F(S) \le (2c + 3 + \log(d_{max}))\mathsf{I}_{\mathsf{C}}(S).$$

Hence, the time complexity of conducting a forward propagation with the SKIP algorithm is bounded by $O(\log(d_{max})\mathsf{I}_{\mathsf{C}}(S))$.

6.2 Heuristic Condition

According to Lemma 1, the expected influence of an arbitrary seed set could be estimated by the RR-set-based method. On the other hand, we could also run many forward propagations and take the average as an estimator of influence. It is natural to ask this question: when should we use the forward propagation approach to estimate its influence if a seed set is given? From Section 6.1, we have learned that a random forward propagation is bound by $O(\mathsf{I}_{\mathsf{C}}(S))$. Intuitively, when the expected influence is low, the forward propagation approach should be used. In this subsection, we first derive an upper bound for the time cost of generating a random RR set under WC model on *undirected* graphs. Then with the aid of this bound, we are allowed to heuristically derive a condition to answer the aforementioned question under WC model.

Upper bound for expected cost of a random RR set. According to [44], the following lemma provides an upper bound on the expected influence of a given seed set.

Lemma 17. *Given a WC model with an undirected graph $G = (V, E)$ and a seed set $S \subseteq V$, we have $\mathsf{I}_{\mathsf{C}}(S) \le |E(S, V\backslash S)| + |S|$, where $E(S, V\backslash S)$ is the set of edges between S and $V\backslash S$.*

The above lemma reveals that the expected influence of a given seed set S is inherently bounded by the size of S plus the total number of the edges incident to S, denoted as $|E(S, V\backslash S)|$. It sets up a connection between the expected influence of a seed set and its degree information. It is highly informative such that we are allowed to further derive an upper bound for the expected cost of generating a random RR set. The derivation is as follows: first, when S includes only one single seed (i.e., $S = \{u\}$), the upper bound in Lemma 17 becomes $d(u) + 1$, where $d(u)$ is the degree of node u; Then plugging it into the inequality (3) in Section 3.2:

$$E_R \le \frac{\theta(V)}{n}\sum_{u\in V}\frac{\theta(d_{in}(u))}{\theta(V)}\mathsf{I}_{\mathsf{C}}(\{u\}) \le \frac{1}{n}\sum_{u\in V}(d(u)+1) = d_{avg} + 1$$

where d_{avg} is the averaged degree over all the nodes in the graph.

Lemma 18. *Given a WC model with an undirected graph $G = (V, E)$, the cost to generate a random RR set on G can be bounded by $d_{avg} + 1$.*

Lemma 18 is remarkable, showing that generating a random RR set under WC model on undirected graphs is computationally efficient, with a cost of only $d_{avg} + 1$.

Heuristic condition. Now we are ready to give a heuristic condition. On one hand, Lemma 17 provides an upper bound (denoted as U) for the expected influence of a given seed set S. Thus according to Theorem 5, the running time cost of a random forward propagation is $O(U)$. On the other hand, according to Lemma 17, the expected cost of generating a random RR set is bounded by $d_{avg} + 1$. Let N_f be the number of forward propagations and N_r be the number of the RR sets. Thus, heuristically we should use the forward propagation approach if it holds that

$$N_f \cdot U \leq N_r \cdot d_{avg}.$$

Here we approximate the cost of a forward propagation to be the upper bound U.

Note that both approaches should be able to provide the similar approximation guarantee. It allows us to establish a relationship between N_f and N_r. We claim that to achieve a similar guarantee, N_f and N_r should satisfy

$$N_f = \frac{\alpha \cdot U}{n} N_r \tag{15}$$

where α is a parameter taking value of 10. The derivation is presented in Section 6.3.

Therefore, by putting together the above results, we draw this condition: we should use the forward propagation approach if the given seed set S satisfies

$$\alpha \cdot U^2 \leq n \cdot d_{avg} = 2m. \tag{16}$$

It implies that if we want to use the forward propagation approach, the total number of the edges incident to the seed set S should be much smaller than the edge number of the graph. It should be pointed out that though this heuristic condition is derived from the properties of undirected graphs, our experiments show it is also effective on directed graphs.

6.3 Theoretical Analysis

In this subsection, we present the derivation of Eq. (15) mentioned in the last subsection. Given a seed set S, let $f_i(S)$ be the number of activated nodes in the i-th forward-propagating simulation starting from S where $i = 1, 2, \ldots, N_f$. Without confusion, we denote it as f_i for simplification. Let U be the deterministic upper bound of influence for the given seed set S according to Lemma 17. Though the expectation $\mathsf{I}_{\mathsf{C}}(S)$ of f_i satisfies $\mathsf{I}_{\mathsf{C}}(S) \leq U$, the value of f_i might be as large as $|V|$, which makes applying the Chernoff bound unattractive.

To tackle this challenge, we define a variable $\hat{f}_i$ by truncating f_i with the threshold $\alpha \cdot U$, that is,

$$\hat{f}_i = \begin{cases} f_i, & \text{if } f_i \leq \alpha U \\ \alpha U, & \text{otherwise} \end{cases} \tag{17}$$

where α is a parameter. We choose $\alpha = 10$ such that the percentage of forward-propagating simulation with influence larger than αU is less that 10%, which comes from the Markov inequality:

$$\Pr[X \geq a] \leq \frac{E[X]}{a},$$

where X is a non-negative random variable and $a > 0$. Let $\hat{\mathsf{I}}_{\mathsf{C}}(S)$ denote the expectation of $\hat{f}_i$. Clearly it holds that $\mathsf{I}_{\mathsf{C}}(S) \geq \hat{\mathsf{I}}_{\mathsf{C}}(S)$.

The Chernoff bound is given as follows:

Lemma 19. *[17] Let $X_1, \ldots, X_n$ be independently distributed in $[0, 1]$. Let $X = \sum_{i=1}^{n} X_i$. Then for $\lambda > 0$,*

$$\Pr[X - E[X] \geq \lambda] \leq \exp\left(-\frac{\lambda^2}{2E[X] + 2/3\lambda}\right).$$

For $\lambda \ddot{Y} E[X]$, then we have $\exp\left(-\frac{\lambda^2}{2E[X]+2/3\lambda}\right) \leq \exp\left(-\frac{\lambda^2}{3E[X]}\right)$. Let N_f be the number of forward-propagating simulations. By applying the Chernoff bound, we have

$$\Pr\left[\sum_{i=1}^{N_f} \frac{\hat{f}_i}{\alpha U} - N_f \frac{\hat{\mathsf{I}}_{\mathsf{C}}(S)}{\alpha U} \geq \lambda_f N_f\right] \leq \exp\left(-\frac{\lambda_f^2}{3\frac{N_f \hat{\mathsf{I}}_{\mathsf{C}}(S)}{\alpha U}}\right).$$

Let $\lambda_f = \frac{\Delta_f N_f}{\alpha U}$. Then it can be rewritten as

$$\Pr\left[\hat{\mathsf{I}}_{\mathsf{C}}(S) \leq \frac{1}{N_f} \sum_{i=1}^{N_f} \hat{f}_i - \Delta_f N_f\right] \leq \exp\left(-\frac{\Delta_f^2 N_f}{3\hat{\mathsf{I}}_{\mathsf{C}}(S)\alpha U}\right). \tag{18}$$

On the other hand, let N_r be the number of RR sets and Y_i be a random variable for the RR set R_i such that $Y_i = 1$ if $S \cap R_i < \emptyset$, and $Y_i = 0$ otherwise. According to Lemma 1 and the Chernoff bound,

$$\Pr\left[\sum_{i=1}^{N_r} Y_i - N_r \frac{\mathsf{I}_{\mathsf{C}}(S)}{n} \geq \lambda_r N_r\right] \leq \exp\left(-\frac{\lambda_r^2}{3N_r \frac{\mathsf{I}_{\mathsf{C}}(S)}{n}}\right),$$

where n is the number of nodes in the graph. Let $\lambda_r = \frac{N_r \Delta_r}{n}$, and then we can rewrite the above inequality as follows,

$$\Pr\left[\mathsf{I}_{\mathsf{C}}(S) \leq \frac{n}{N_r} \sum_{i=1}^{N_r} Y_i - \Delta_r N_r\right] \leq \exp\left(-\frac{N_r \Delta_r^2}{3n \cdot \mathsf{I}_{\mathsf{C}}(S)}\right). \tag{19}$$

Assume that the parameter $\alpha = 10$ is a favorable setting such that $\hat{\mathsf{I}}_{\mathsf{C}}(S)$ is close to $\mathsf{I}_{\mathsf{C}}(S)$, (i.e., $\hat{\mathsf{I}}_{\mathsf{C}}(S) \approx \mathsf{I}_{\mathsf{C}}(S)$). Then taking a comparison between (18) and (19), to achieve a similar lower bound (i.e., $\Delta_f \approx \Delta_r$) with the same failure probability (i.e., $\exp\left(-\frac{\Delta_f^2 N_f}{3\hat{\mathsf{I}}_{\mathsf{C}}(S)\alpha U}\right) = \exp\left(-\frac{N_r \Delta_r^2}{3n \cdot \mathsf{I}_{\mathsf{C}}(S)}\right)$), we have

$$\frac{N_f}{\alpha U} = \frac{N_r}{n}.$$

Therefore, we finally obtain Eq. (15).

7 ADDITIONAL RELATED WORK

There has been a large body of research on IM, e.g., [13–16, 18, 20, 23–25, 29, 30, 33, 34, 36, 43, 51], in the literature. Kempe et al. [30] present the first seminal work on IM, and show that finding k users that maximize the influence is NP-hard. They provide a greedy algorithm that provides $(1 - 1/e - \epsilon)$-approximate solution, which requires $\Omega(k \cdot m \cdot n \cdot poly(1/\epsilon))$ running time, and is too expensive on large social networks. A plethora of research works, e.g., [7, 14–16, 20, 24, 25, 29, 43], study how to improve the efficiency of the IM problem. Most algorithms are heuristic and fail to provide approximation guarantee. The states of the art are the RR set based solutions [9, 42, 46–48], as discussed in Section 2.2, which provide superb efficiency and a strong theoretical guarantee.

Besides, a plethora of research work focuses on more practical scenarios rather than the classic IM. For instance, topic-aware IM, by taking consideration of the topic propagated, is studied by [34, 38]. Time-aware IM, which considers a time constraint during the diffusion process, is studied in [21, 35]. Competitive IM [12, 36] considers the scenarios where several competitors spread their influences in the same social networks simultaneously and their diffusion interferes with each other. There also exist studies on IM under budget constraints [8, 38], constraint to user groups [49], and under adaptive settings [27, 45]. These are orthogonal to our study.

Table 2. Summary of datasets (K = 10^3, M = 10^6, B = 10^9)

Dataset	Type	n	m
Pokec	directed	1.6M	30.6M
Orkut	undirected	3.1M	117.2M
Twitter	directed	41.7M	1.5B
Friendster	undirected	65.6M	1.8B

Influence estimation with accuracy guarantee is also studied in the literature, e.g., [37, 41]. Lucier et al. [37] introduce the influence estimation method with accuracy guarantee, INFEST, which is tailored for distributed settings. Nguyen et al. [41] propose the SIEA method to estimate the influence spread with accuracy guarantee, which is built on two components: the important sampling method IICP that only samples non-singular cascades, and the estimation method RSA that can reduce the number of cascade samples required for desired accuracy guarantee.

8 EXPERIMENTS

This section evaluates our solutions against alternatives. All experiments are conducted on a Linux machine with an Intel Xeon CPU clocked at 2.70GHz and 350 GB memory. We implement all of our algorithms in C++ and compile all algorithms with full optimization. We repeat each algorithm five times and report the average running time as the query performance.

Algorithms. For evaluating the effectiveness of SUBSIM, we compare our solutions against the three state-of-the-art solutions, IMM, SSA and OPIM-C, which all adopt the vanilla RR set generation algorithm (Algorithm 2). The C++ implementations of these solutions are available at [4], [6] and [5], respectively. For our solution, we first implement based on the existing state-of-the-art OPIM-C and integrate our SUBSIM framework for RR set generation. Moreover, we also integrate the OPIM-C with the RR set generation method SKIS [40] for performance comparison, and name it *OPIM-C+SKIS* to be distinguished from the OPIM-C with the vanilla RR set generation algorithm. The algorithm SKIS is implemented according to the pseudocode in [40].

For evaluating the effectiveness of HIST under highly influential scenarios, we implement two versions of HIST, one with vanilla RR set generation algorithm and one with SUBSIM framework for RR set generation. To compare HIST with non-RR-set-based solutions, we include the snapshot-based method PMC [43] as a competitor, whose C++ implementation can be found at [2].

In the experiments for forward propagation issues, we implement two versions of the forward propagation, one with SKIP algorithm, and one with the vanilla subset sampling algorithm (that is, flipping a biased coin once for each out-going edge). Besides, to evaluate the effectiveness of the heuristic condition proposed in Section 6.2, we implement three estimators for influence estimation. The first one is the forward propagation estimator with the SKIP sampling method (i.e., we conduct the IC propagation simulations using the SKIP sampling method many times, and then take the average), dubbed as *FP-SKIP*; the second one is the forward propagation estimator with the IICP sampling method [41], dubbed as *FP-IICP*; the third one is the RR-set-based estimator, dubbed as *RRSE*, which samples sufficient RR sets and then estimate the influence spread according to Lemma 1. The IICP sampling method is implemented according to the pseudocode in [41].

Datasets. We evaluate our experiments on four benchmark datasets that are publicly available at [1, 3]. The summary of these datasets is shown in Table 2.

Parameter Settings. Recap that all the algorithms include an error parameter ϵ and a failure probability parameter δ. Following previous work [46], we set $\epsilon = 0.1$ and $\delta = 1/n$ for all solutions in the experiments. To examine the effectiveness of our SUBSIM, we compare our SUBSIM against the vanilla RR set generation algorithm under IC model with different distribution settings. We

first test on WC model, where the weight of an edge (w, u) is set as $1/d_{in}(u)$. Then we test the case when the weight of edges follows skewed distributions, in particular, exponential distribution and Weibull distribution. For exponential distribution, the *probability density function (PDF)* is $f(x) = \lambda e^{-\lambda x}$. We set $\lambda = 1$ and sample the weight of each edge with this setting. For each node v, we scale the sum of the weights of its incoming edges to 1. For Weibull distribution, the PDF is $f(x) = \frac{a}{b} \cdot \left(\frac{x}{b}\right)^{a-1} \cdot e^{-(x/b)^a}$. Following previous studies [47], we sample a and b from $[0, 10]$ uniformly at random for each edge e. For each node v, we scale the sum of the probability weight of its incoming edges to 1.

We then examine the effectiveness of HIST in high influence networks, where the average size of random RR sets tends to be quite large. We design our experiments by varying the average size of random RR sets under two settings. The first setting, dubbed as *WC variant*, is similar to WC model except that we introduce a constant $\theta \geq 1$ such that the weight of an edge (w, u) is set as $\min\{1, \theta/d_{in}(u)\}$. By changing θ, we are able to vary the average size of random RR sets. We then vary θ on each dataset such that the average size of random RR sets is approximately $\{50, 400, 1000, 4000, 8000, 32000\}$. We denote the setting as $\theta_{50}, \theta_{400}, \theta_{1K}, \theta_{4K}, \theta_{8K}, \theta_{32K}$, respectively. The second setting is the *Uniform IC* setting where all edges have the same weight p. We vary the weight p on each dataset such that the average size of random RR sets is approximately $\{50, 400, 1000, 4000, 8000, 32000\}$. We denote the setting as $p_{50}, p_{400}, p_{1K}, p_{4K}, p_{8K}, p_{32K}$, respectively. As for the baseline PMC [43], we set the parameter R (i.e., the number of snapshots) as 200, following the default setting in [43].

Accuracy Measure. To measure estimation accuracy when evaluating different influence estimators, we compute the relative error and then take the average, which is expressed as $\frac{1}{\theta} \sum_{i=1}^{\theta} |\hat{x}_i/x_i - 1|$, where $\hat{x}_i$ and x_i is the estimated and exact influence of the i-th seed set, respectively. The exact expected influence is computed with guarantee that relative error is less than 0.1%.

8.1 Effectiveness of SUBSIM

In the first set of experiments, we examine the effectiveness of SUBSIM against IMM, SSA, OPIM-C, and OPIM-C+SKIS under WC setting. Figure 1 reports the average running time on the four datasets. The first observation is that SUBSIM consistently outperforms alternatives on all the tested datasets. Compared to OPIM-C, even though we only modify the RR set generation algorithm, SUBSIM is still up to 15x faster than OPIM-C on Twitter. SUBSIM further outperforms SSA (resp. IMM) by up to an order (resp. three orders) of magnitude. Compared with OPIM-C+SKIS which combines OPIM-C and SKIS, our SUBSIM also achieves a better performance by up to 20x on Twitter. It can be explained with the fact that our SUBSIM optimizes the generation of both the singular and non-singular RR-sets. The cost of singular RR set generation in SUBSIM is also much smaller than the vanilla RR set generation algorithm which takes an expected cost proportional to the average degree. For example, on the WC model, the cost of singular RR set generation takes only $O(1)$ cost with our SUBSIM. Note that OPIM-C+SKIS fails to consistently beat the vanilla OPIM-C. To explain, the greedy algorithm in SKIS (see Alg. 3 in [40]) iteratively selects a node $\hat{v}$ having the largest marginal influence gain which is float data type due to the importance sampling mechanism, and is implemented using max heap. In contrast, OPIM-C with the vanilla RR set generation iteratively pick a node $\hat{v}$ with maximal marginal coverage which is represented with an integer value, and thus could be implemented more efficiently with inverted list and lazy update.

In the second set of experiments, we consider the skewed distribution settings, i.e., when the edges follow the exponential or Weibull distribution. We omit the results for IM algorithms since the experimental result follows a similar trend. Instead, we focus on comparing the cost of the vanilla RR set generation algorithm, denoted as *vanilla*, with that of our SUBSIM for RR set generation.

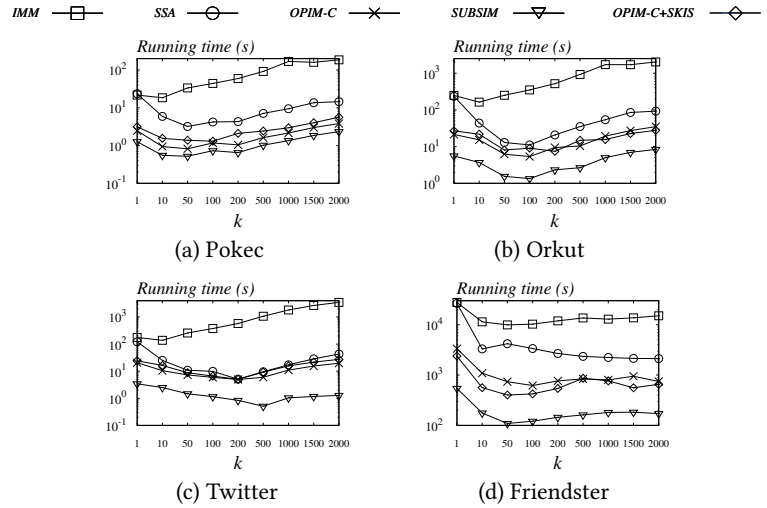


Fig. 1. Varying k: Running time of IM algorithms under WC model.

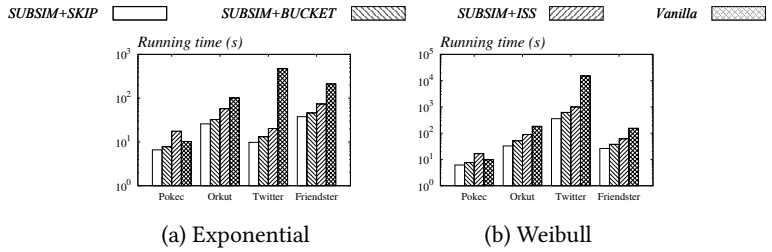
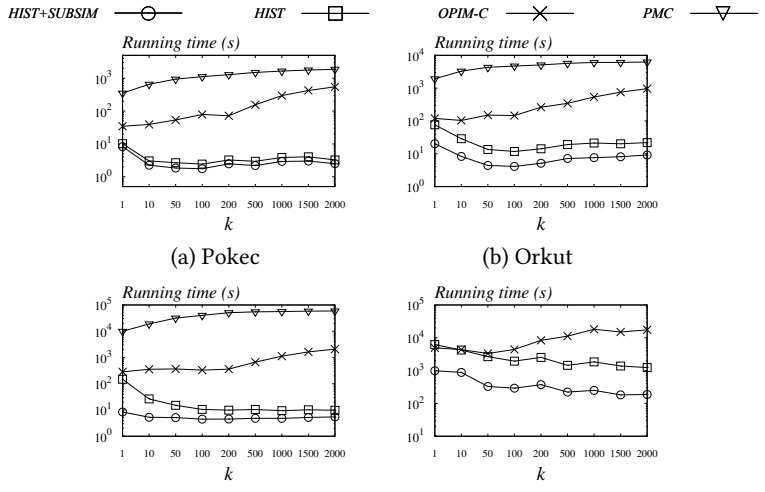


Fig. 2. Skewed distribution: RR set generation cost.

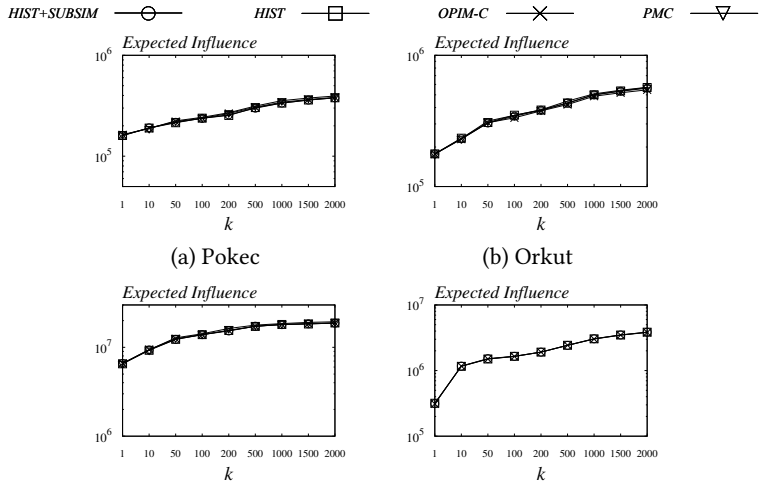
We implement SUBSIM with BUCKET with $\beta = 2$ (denoted as *SUBSIM + BUCKET*), ISS (denoted as *SUBSIM + ISS*), and SKIP (denoted as *SUBSIM + SKIP*), respectively. We generate $2^{10} \times 1000$ random RR sets with each of the four algorithms on each dataset, and report their running time. As shown in Figure 2, SUBSIM + SKIP consistently keeps its advantage on all four tested datasets and achieves up to 41x (resp. 43x) speedup over vanilla under exponential (resp. Weibull) distribution. Besides, even though the ISS sampling method has the optimal time complexity of $O(1 + \mu)$ for the subset sampling problem, however, SUBSIM+SKIP gives a better practical performance than SUBSIM+ISS, and achieves up to 2.7x (resp. 2.8x) speedup under exponential (resp. Weibull) distribution. This result can be explained by the fact that the ISS sampling method takes a two-dimensional index structure, which is complicate and may hamper the cache efficiency. Compared with the vanilla RR set generation algorithm, the ISS algorithm is still faster on three (Orkut, Twitter, and Friendster) out of four tested datasets, and is up to an order of magnitude faster on Twitter datasets. Furthermore, SUBSIM+SKIP consistently outperforms over SUBSIM+BUCKET on all four graphs, and achieves by up to 1.4x (resp. 1.7x) speedup under exponential (resp. Weibull) distribution, which agrees with the theoretical analysis that SKIP is optimal for the index-free sorted subset sampling problem.



(a) Pokec (b) Orkut

(c) Twitter (d) Friendster

Fig. 3. Varying k: Running time under WC variant setting.



(a) Pokec (b) Orkut

(c) Twitter (d) Friendster

Fig. 4. Varying k: Expected influence under WC variant setting.

8.2 Effectiveness of HIST

Our first set of experiments examines the performance of our HIST when k varies under WC variant setting. We fix θ and set it to θ_{4K} for each dataset, and vary k with $\{1, 10, 50, 100, 200, 500, 1000, 1500, 2000\}$. Figure 3 shows the average running time of HIST (with vanilla RR set generation algorithm), HIST+SUBSIM (with SUBSIM for RR set generation), OPIM-C, and PMC. We observe that with the increase of size k, the benefit of applying our HIST algorithm becomes more significant, and HIST is at least an order of magnitude faster than OPIM-C. HIST+SUBSIM further achieves up to an order of magnitude speedup over HIST since HIST+SUBSIM adopts SUBSIM for RR set generation. We omit the result of PMC on dataset Friendster since it runs

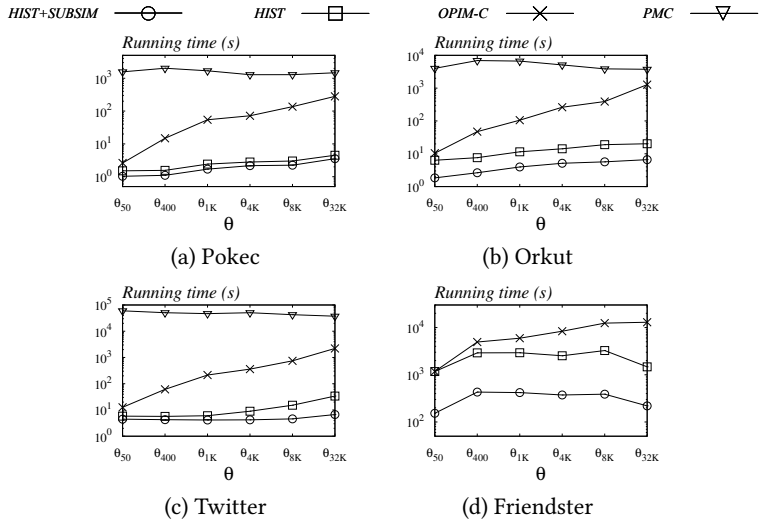


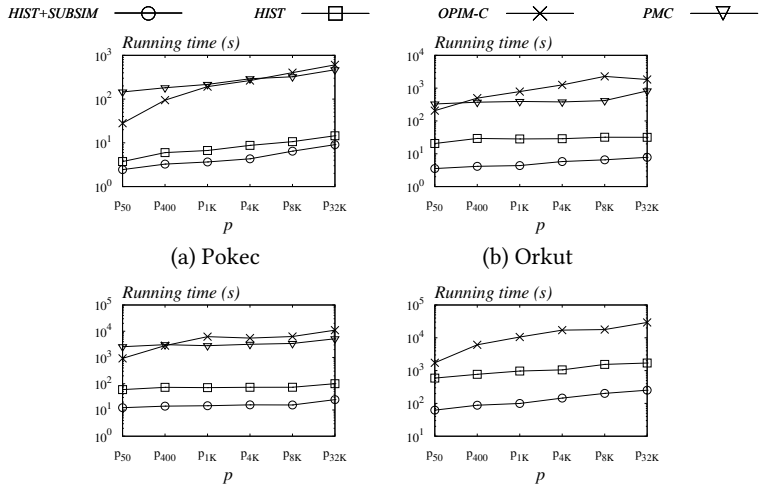
Fig. 5. Varying θ: Running time under WC variant setting.

out of memory on our machine (due to the same reason, we also omit the result of PMC in the later experiments). Note that the RR-set-based solutions outperform PMC by a large margin among all three datasets under WC variant model. With the size k of the seed set increasing, the gap between OPIM-C and PMC gets smaller. In contrast, this situation does not happen to our solutions HIST and HIST+SUBSIM. Figure 4 shows the expected influence when we increase k from 1 to 2000 with θ_{4k} setting. The expected influence gains a significant increase when we increase k from 1 to 2000 on all four datasets.

In our second set of experiments, we vary the average size of random RR sets under WC variant setting. We fix $k = 200$ and vary θ with $\theta_{50}, \theta_{400}, \theta_{1K}, \theta_{4K}, \theta_{8K}, \theta_{32K}$ on each dataset. Figure 5 shows the running time of our solutions against OPIM-C. We can observe that even when the average size of random RR sets is around 50, our HIST is already as competitive as OPIM-C. When the size of random RR sets further increases, HIST shows a more significant advantage and is up to two orders of magnitude faster than OPIM-C when $\theta = \theta_{32K}$. Besides, HIST+SUBSIM is always two orders of magnitude faster than OPIM-C when $\theta = \theta_{32K}$. Compared with PMC, the RR-set-based solutions OPIM-C, HIST, and HIST+SUBSIM are faster. Observe that as the average size of random RR sets is gradually increasing, the advantage of HIST/HIST+SUBSIM over PMC keeps relatively stable, whereas the curve of OPIM-C becomes closer to that of PMC.

In our third set of experiments, we vary the average size of random RR sets under Uniform IC setting. We fix $k = 200$ and vary p with $\{p_{50}, p_{400}, p_{1K}, p_{4K}, p_{8K}, p_{32K}\}$. Figure 6 shows the running time of all three algorithms. We can see that even when the average size of RR sets is around 50, HIST is already several times faster than OPIM-C. When p is set to p_{32K}, HIST (resp. HIST+SUBSIM) is at least an order (resp. two orders) of magnitude faster than OPIM-C. Our HIST and HIST-SUBSIM also maintain a significant advantage compared with PMC. Note that under Uniform IC model, when p is large (say larger than p_{400}), PMC gives a better performance than OPIM-C on datasets Orkut and Twitter. We also examine the effectiveness of our solutions when k varies under Uniform IC setting. The result is similar to our findings under WC variant setting and is omitted.

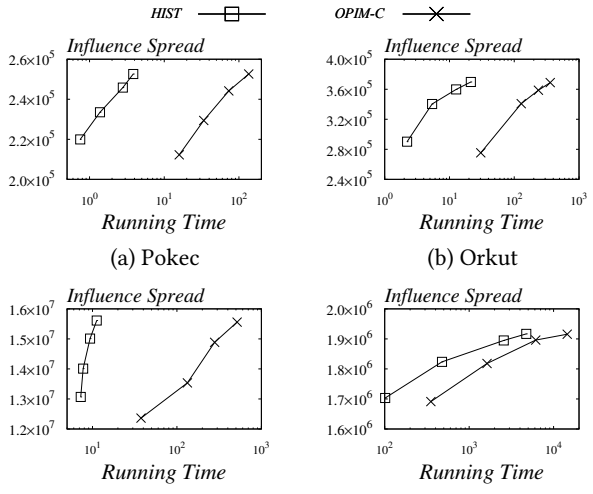
In our forth set of experiments, we run HIST (without SUBSIM) and OPIM-C with various error thresholds ϵ under WC variant model. We fix $k = 200$, $\theta = \theta_{4K}$, and vary ϵ with $\{0.5, 0.25, 0.1, 0.01\}$.



(a) Pokec (b) Orkut

(c) Twitter (d) Friendster

Fig. 6. Varying p: Running time under Uniform IC setting.



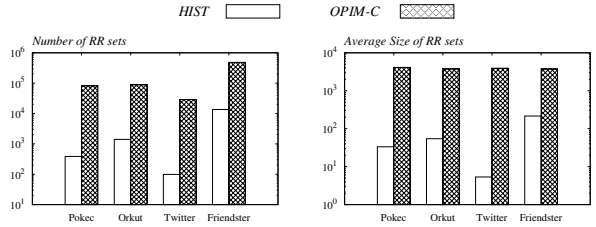
(a) Pokec (b) Orkut

(c) Twitter (d) Friendster

Fig. 7. Varying ϵ: Running time vs Empirical influence under WC variant setting.

Figure 7 shows the running time and its empirical influence. Observe that when achieving a roughly identical empirical influence, HIST takes a much smaller running time compared with OPIM-C. It implies that HIST does improve the trade-off between the running time and empirical influence.

In the fifth set of experiments, we report some statistics of RR sets with our HIST under WC variant setting with $k = 2000$ and $\theta = \theta_{4K}$. Figure 8(a) reports the number of RR sets generated in the sentinel set selection phase of HIST. We compare with the number of RR sets generated by OPIM-C and we observe that the number of random RR sets required by our HIST is two orders of magnitude smaller than that required by OPIM-C in most datasets. Figure 8(b) reports the average size of random RR sets generated by our HIST against OPIM-C. Observe that the average size of



(a) Number of RR sets (b) Average size of RR sets

Fig. 8. Statistics of RR sets.

Table 3. Average size of RR sets with the sentinel set

Alg \ Dataset	Pokec	Orkut	Twitter	Friendster
Greedy-Degree	39.8	61.1	18.5	197.0
Greedy-Cost	78.8	134.2	57.01	324.1
standard greedy	82.7	137.3	32.2	404.3

random RR sets with HIST is reduced by up to 700x. To explain, when a node in the sentinel set is met, the RR set generation with HIST can immediately stop, reducing the size of RR sets.

In the last set of experiments, we examine the effectiveness of our revised greedy algorithm, Greedy-Degree (see Alg. 9), against standard greedy and Greedy-Cost (see Appendix B) under WC variant seting with $\theta = \theta_{4K}$ and $k = 2000$. The experiment is designed as follows: we generate a collection $\mathcal{R}$ of RR sets, then respectively apply these three greedy algorithms to get a sentinel set, and finally report the average size of RR sets with each obtained sentinel set. Since the average size of RR sets with a sentinel set might be affected by the size of the sentinel set, we fix $b = 200$, which is roughly equal to the size of S_b^* obtained by our HIST when $k = 2000$. Besides, because a larger $\mathcal{R}$ can improve the quality of the sentinel set, thus for fair comparison, we fix the size of $\mathcal{R}$ on each dataset. We set the size of $\mathcal{R}$ as 1000, 1000, 100, and 10000 on datasets Pokec, Orkut, Twitter, and Friendster, respectively, such that it is similar as the number used in the sentinel set selection phase of HIST which is reported in Figure 8(a). We repeat the experiment 20 times and take the average. Table 3 reports the average size of RR sets with the sentinel set S_{200}^*. We observe that the average size of the RR sets with the sentinel set obtained by Greedy-Degree is much smaller, compared with its competitors, standard greedy and Greedy-Cost. Note that Greedy-Cost gives a smaller average size of RR sets than standard greedy, except on Twitter. It is explained that the performance of Greedy-Cost relies on the size of $\mathcal{R}$, and a larger $\mathcal{R}$ can better capture the probability space of RR sets, while on dataset Twitter, the number of RR sets is only 100.

In summary, the result indicates that the larger the average size of random RR sets are, the more effective our HIST and HIST+SUBSIM are. In high influence networks, the average size of random RR sets tends to be large and our proposed solutions are preferred choices.

8.3 Forward Propagation

In the first set of experiments, we examine the effectiveness of SKIP in forward propagation under WC setting. We first invoke SUBSIM with $k = 10, 50, 100$ to get a seed set S, and then run each version (i.e., SKIP and vanilla) of the forward propagation starting from S 1000 times. We repeat this process 10 times and take the average. Figure 9 shows the running time of forward propagations on four datasets. We can see that on all four datasets, the SKIP version of the forward propagation consistently run faster than the vanilla version. Specifically, the SKIP version yields approximately

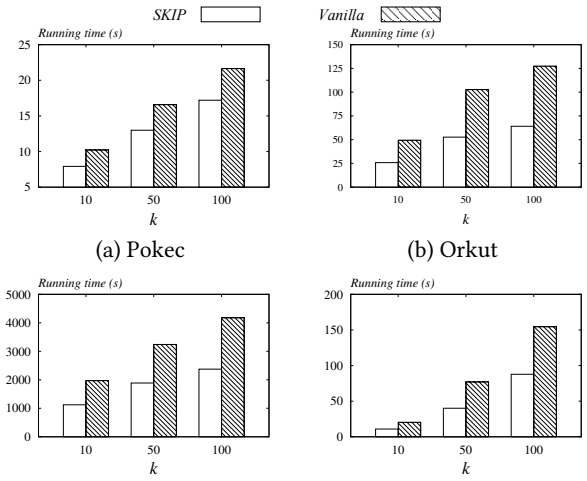


Fig. 9. Running time of forward propagations.

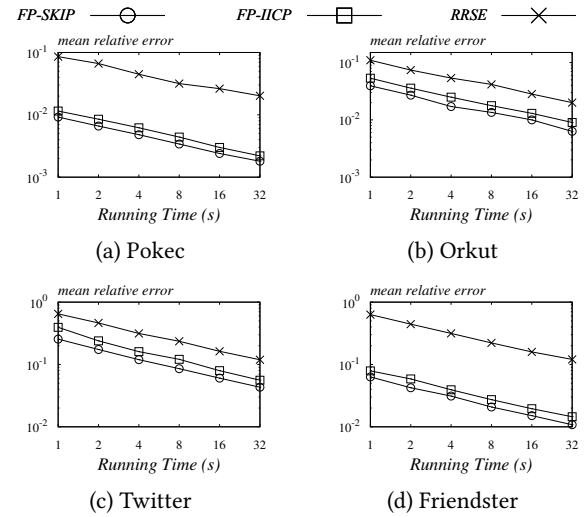


Fig. 10. Varying *Running time*: mean relative error of three influence estimators.

2x speed up on the datasets Orkut, Twitter and Friendster. Note that the performance of the SKIP algorithm in forward propagation is not as remarkble as that in RR set generation, which is up to 43x (see Figure 2). It could be explained that the forward propagation process visits a large number of nodes with low degree, which is unfavourable for SKIP.

In the second set of experiments, to evaluate the effectiveness of our proposed condition (see Inequality (16)) under WC setting, we randomly generate 200 seed sets, each of which has 100 seeds and satisfies the heuristic condition. We vary running time with $\{1, 2, 4, 8, 16, 32\}$ seconds, and then compare the estimation accuracy of FP-SKIP, FP-IICP, and RRSE. Figure 10 reports their mean relative error performance. Note that both forward-propagation-based estimators (i.e., FP-SKIP and FP-IICP) demonstrate better estimation accuracy than RRSE on all four datasets. In particular, on Pokec (Friendster, resp.), FP-SKIP achieves a mean relative error of 1.0% (6.9%, resp.) with only one second running cost, whereas RRSE has to take 32 seconds to get a inferior score, 2.4% (12%, resp.).

In spite of the fact that the heuristic condition is obtained from undirected graphs, the experimental results demonstrate that it can also be applied to directed graphs. Furthermore, our FP-SKIP gives a more accurate estimation since the SKIP sampling method is efficient such that we can generate more simulations with the same running time, compared with the IICP method.

9 CONCLUSION

This paper presents SUBSIM, an efficient framework for RR set generation. We develop an index-free algorithm SKIP for the sorted subset sampling problem. We further present HIST to further tackle the challenging scalability issues in high influence networks. Furthermore, we present the time complexity of the forward propagation and a heuristic condition for estimator choice.

REFERENCES

[1] 2013. KONECT Datasets. http://konect.uni-koblenz.de/.
[2] 2014. PMC code. https://github.com/todo314/pruned-monte-carlo.
[3] 2014. SNAP Datasets. http://snap.stanford.edu/data.
[4] 2015. IMM code. https://sourceforge.net/projects/im-imm/.
[5] 2017. OPIM-C code. https://github.com/tangj90/OPIM.
[6] 2017. SSA code. https://github.com/hungnt55/Stop-and-Stare.
[7] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. 2017. Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study. In *SIGMOD*. 651–666.
[8] Song Bian, Qintian Guo, Sibo Wang, and Jeffrey Xu Yu. 2020. Efficient Algorithms for Budgeted Influence Maximization on Massive Social Networks. *Proc. VLDB Endow.* 13, 9 (2020), 1498–1510.
[9] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *SODA*. 946–957.
[10] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.
[11] Karl Bringmann and Konstantinos Panagiotou. 2017. Efficient Sampling Methods for Discrete Distributions. *Algorithmica* 79, 2 (2017), 484–508.
[12] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. In *WWW*. 665–674.
[13] Shuo Chen, Ju Fan, Guoliang Li, Jianhua Feng, Kian-Lee Tan, and Jinhui Tang. 2015. Online Topic-Aware Influence Maximization. *PVLDB* 8, 6 (2015), 666–677.
[14] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*. 1029–1038.
[15] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *SIGKDD*. 199–208.
[16] Suqi Cheng, Huawei Shen, Junming Huang, Wei Chen, and Xueqi Cheng. 2014. IMRank: influence maximization via finding self-consistent ranking. In *SIGIR*. 475–484.
[17] Fan R. K. Chung and Lincoln Lu. 2006. Survey: Concentration Inequalities and Martingale Inequalities: A Survey. *Internet Math.* 3, 1 (2006), 79–127.
[18] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. 2014. Sketch-based Influence Maximization and Computation: Scaling up with Guarantees. In *CIKM*. 629–638.
[19] Paul Dagum, Richard M. Karp, Michael Luby, and Sheldon M. Ross. 1995. An Optimal Algorithm for Monte Carlo Estimation (Extended Abstract). In *FOCS*. 142–149.
[20] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic Influence Maximization: Combining Scalability and Efficiency with Opinion-Aware Models. In *SIGMOD*. 743–758.
[21] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. 2011. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML*. 561–568.
[22] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2010. Learning influence probabilities in social networks. In *WSDM*. 241–250.
[23] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2011. A Data-Based Approach to Social Influence Maximization. *PVLDB* 5, 1 (2011), 73–84.
[24] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. 2011. CELF++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW*. 47–48.
[25] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. 2011. SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model. In *ICDM*. 211–220.

[26] Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. 2020. Influence Maximization Revisited: Efficient Reverse Reachable Set Generation with Bound Tightened. In *SIGMOD*. ACM, 2167–2181.

[27] Kai Han, Keke Huang, Xiaokui Xiao, Jing Tang, Aixin Sun, and Xueyan Tang. 2018. Efficient Algorithms for Adaptive Influence Maximization. *PVLDB* 11, 9 (2018), 1029–1040.

[28] Keke Huang, Sibo Wang, Glenn S. Bevilacqua, Xiaokui Xiao, and Laks V. S. Lakshmanan. 2017. Revisiting the Stop-and-Stare Algorithms for Influence Maximization. *PVLDB* 10, 9 (2017), 913–924.

[29] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. IRIE: Scalable and Robust Influence Maximization in Social Networks. In *ICDM*. 918–923.

[30] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*. 137–146.

[31] Donald Ervin Knuth. 1997. *The art of computer programming*. Vol. 3.

[32] Andreas Krause and Daniel Golovin. 2014. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 71–104.

[33] Siyu Lei, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart. 2015. Online Influence Maximization. In *SIGKDD*. 645–654.

[34] Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. 2015. Real-time Targeted Influence Maximization for Online Advertisements. *PVLDB* 8, 10 (2015), 1070–1081.

[35] Bo Liu, Gao Cong, Dong Xu, and Yifeng Zeng. 2012. Time constrained influence maximization in social networks. In *ICDM*. 439–448.

[36] Wei Lu, Wei Chen, and Laks V. S. Lakshmanan. 2015. From Competition to Complementarity: Comparative Influence Diffusion and Maximization. *PVLDB* 9, 2 (2015), 60–71.

[37] Brendan Lucier, Joel Oren, and Yaron Singer. 2015. Influence at scale: Distributed computation of complex contagion in networks. In *SIGKDD*. 735–744.

[38] Hung T. Nguyen, Thang N. Dinh, and My T. Thai. 2016. Cost-aware Targeted Viral Marketing in billion-scale networks. In *INFOCOM*. 1–9.

[39] Hung T. Nguyen, Thang N. Dinh, and My T. Thai. 2018. Revisiting of 'Revisiting the Stop-and-Stare Algorithms for Influence Maximization'. In *CSoNet*. 273–285.

[40] Hung T. Nguyen, Tri P. Nguyen, NhatHai Phan, and Thang N. Dinh. 2017. Importance Sketching of Influence Dynamics in Billion-Scale Networks. In *ICDM*. 337–346.

[41] Hung T. Nguyen, Tri P. Nguyen, Tam N. Vu, and Thang N. Dinh. 2017. Outward Influence and Cascade Size Estimation in Billion-scale Networks. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1 (2017), 20:1–20:30.

[42] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 2016. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In *SIGMOD*. 695–710.

[43] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2014. Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations. In *AAAI*. 138–144.

[44] Grant Schoenebeck and Biaoshuai Tao. 2020. Influence Maximization on Undirected Graphs: Toward Closing the (1-1/e) Gap. *ACM Trans. Economics and Comput.* 8, 4 (2020), 22:1–22:36.

[45] Jing Tang, Keke Huang, Xiaokui Xiao, Laks V. S. Lakshmanan, Xueyan Tang, Aixin Sun, and Andrew Lim. 2019. Efficient Approximation Algorithms for Adaptive Seed Minimization. In *SIGMOD*. 1096–1113.

[46] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online Processing Algorithms for Influence Maximization. In *SIGMOD*. 991–1005.

[47] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *SIGMOD*. 1539–1554.

[48] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*. 75–86.

[49] Rajan Udwani. 2018. Multi-objective Maximization of Monotone Submodular Functions with Cardinality Constraint. In *NeurIPS*. 9513–9524.

[50] Alastair J. Walker. 1977. An Efficient Method for Generating Discrete Random Variables with General Distributions. *ACM Trans. Math. Softw.* 3, 3 (1977), 253–256.

[51] Yanhao Wang, Qi Fan, Yuchen Li, and Kian-Lee Tan. 2017. Real-Time Influence Maximization on Dynamic Social Streams. *PVLDB* 10, 7 (2017), 805–816.

APPENDIX

A PROOF OF THEOREM 4

We introduce some notations and lemmas that are useful to prove Theorem 4. Denote $\bar{\mu}$ as the total expected number of elements in T checked by SKIP to determine whether x_j is added to S (Lines 9-10). Thus, SKIP takes $O(1+\bar{\mu})$ time in expectation, where "1" is from the stopping step of while loop when it meets the condition $j \ i \ h$ (Line 7).

Similarly, let $\bar{\mu}(p_i, \ldots, p_j)$ denote the expected number of elements checked when $T = \{x_i, \ldots, x_j\}$, e.g., $\bar{\mu} = \bar{\mu}(p_1, \ldots, p_h)$. Consider that SKIP performs on $\{x_i, \ldots, x_j\}$ and x_k is the first element checked. The probability for such a case is $(1-p_i)^{k-i}p_i$, as $\Pr[X = k-i+1]$ follows the geometric distribution $G(p_i)$. After checking x_k, SKIP performs on $\{x_{k+1}, \ldots, x_j\}$. Thus, by Markov chain,

$$\bar{\mu}(p_i, \ldots, p_j) = \tilde{\mathcal{O}}_{k=i} (1-p_i)^{k-i}p_i \cdot \ 1 + \bar{\mu}(p_{k+1}, \ldots, p_j) \quad .$$

The following lemma shows the monotonicity of $\bar{\mu}(p_i, \ldots, p_j)$.

Lemma 20. $\bar{\mu}(p_{i+1}, \ldots, p_j) \le \bar{\mu}(p_i, \ldots, p_j)$.

Proof. We prove the lemma by induction. When $i = j-1$, we have $\bar{\mu}(p_{i+1}) = p_{i+1}$ and $\bar{\mu}(p_i, p_{i+1}) = p_i + (1-p_i)p_i + p_i p_{i+1} = (2 + p_{i+1} - p_i)p_i$, which indicates that $\bar{\mu}(p_{i+1}) \le \bar{\mu}(p_i, p_{i+1})$. Assume that for any $i \ge i^*$, it holds that

$$\bar{\mu}(p_{i+1}, \ldots, p_j) \le \bar{\mu}(p_i, \ldots, p_j).$$

Now consider $i = i^* - 1$. Similar to (A), we have

$$\bar{\mu}(p_{i+1}, \ldots, p_j) = \tilde{\mathcal{O}}_{k=i+1} (1-p_{i+1})^{k-i-1}p_{i+1} \cdot \ 1 + \bar{\mu}(p_{k+1}, \ldots, p_j) \quad .$$

For any $\ell = i, \ldots, j$, define

$$\Delta_\ell := \tilde{\mathcal{O}}_{k=i} (1-p_i)^{k-i}p_i - \tilde{\mathcal{O}}_{k=i+1} (1-p_{i+1})^{k-i-1}p_{i+1} = (1-p_{i+1})^{\ell-i} - (1-p_i)^{\ell-i+1}.$$

It is easy to verify that $\Delta_\ell \ge 0$ for any $\ell = i, \ldots, j$ owing to the fact that $\{p_i, \ldots, p_j\}$ are in non-ascending order. Besides, the following conclusion holds by definition, and will be used later,

$$\Delta_{\ell+1} + p_{i+1}(1-p_{i+1})^{l-i} = \Delta_\ell + p_i(1-p_i)^{l-i+1}. \tag{20}$$

Then, we have

$$\begin{aligned}
\bar{\mu}(p_i, \ldots, p_j) &= \tilde{\mathcal{O}}_{k=i} (1-p_i)^{k-i}p_i \cdot \ 1 + \bar{\mu}(p_{k+1}, \ldots, p_j) \\
&= p_i \ \ 1 + \bar{\mu}(p_{i+1}, \ldots, p_j) \ \ + \tilde{\mathcal{O}}_{k=i+1} (1-p_i)^{k-i}p_i \cdot \ 1 + \bar{\mu}(p_{k+1}, \ldots, p_j) \\
&\ge \Delta_{i+1} \ \ 1 + \bar{\mu}(p_{i+2}, \ldots, p_j) \ \ + p_{i+1} \ \ 1 + \bar{\mu}(p_{i+2}, \ldots, p_j) \ \ + \tilde{\mathcal{O}}_{k=i+2} (1-p_i)^{k-i}p_i \cdot \ 1 + \bar{\mu}(p_{k+1}, \ldots, p_j) \quad .
\end{aligned}$$

The inequality is due to $\bar{\mu}(p_{i+1}, \ldots, p_j) \ge \bar{\mu}(p_{i+2}, \ldots, p_j)$ and $\Delta_{i+1} + p_{i+1} = p_i + (1-p_i)p_i$ by (20).

Recursively, we have

$$\begin{aligned}\bar{\mu}(p_i,\dots,p_j) &\geq \Delta_{i+2}\ \ 1+\bar{\mu}(p_{i+3},\dots,p_j)\ \ + \sum_{k=i+1}^{i+2}(1-p_{i+1})^{k-i-1}p_{i+1}\cdot\ \ 1+\bar{\mu}(p_{k+1},\dots,p_j)\\ &\quad + \sum_{k=i+3}^{j}(1-p_i)^{k-i}p_i\cdot\ \ 1+\bar{\mu}(p_{k+1},\dots,p_j)\\ &\geq \cdots \geq \Delta_j + \bar{\mu}(p_{i+1},\dots,p_j).\end{aligned}$$

Note that $\Delta_j \geq 0$ as $p_i \geq p_{i+1}$, which immediately concludes the lemma.

Lemma 21. *For any $p_k \leq p'_k$, $\bar{\mu}(p_i,\dots,p_k,\dots,p_j) \leq \bar{\mu}(p_i,\dots,p'_k,\dots,p_j)$.*

Proof. We prove the lemma by induction. When $i = j = k$, it obviously holds that $\bar{\mu}(p_k) = p_k \leq p'_k = \bar{\mu}(p'_k)$. Assume that $\bar{\mu}(p_i,\dots,p_k,\dots,p_j) \leq \bar{\mu}(p_i,\dots,p'_k,\dots,p_j)$ that for any $i \geq i^*$. Now consider the following two cases when $i = i^* - 1$.

Case (i) $k > i$. With assumption that for any $\ell + 1 \geq i + 1 \geq i^*$, $\bar{\mu}(p_{\ell+1},\dots,p_k,\dots,p_j) \leq \bar{\mu}(p_{\ell+1},\dots,p'_k,\dots,p_j)$, then we have:

$$\begin{aligned}\bar{\mu}(p_i,\dots,p_k,\dots,p_j) &= \sum_{\ell=i}^{j}(1-p_i)^{\ell-i}p_i\cdot\ \ 1+\bar{\mu}(p_{\ell+1},\dots,p_k,\dots,p_j)\\ &\leq \sum_{\ell=i}^{j}(1-p_i)^{\ell-i}p_i\cdot\ \ 1+\bar{\mu}(p_{\ell+1},\dots,p'_k,\dots,p_j) \quad = \bar{\mu}(p_i,\dots,p'_k,\dots,p_j).\end{aligned}$$

Case (ii) $k = i$. Let $\Gamma_\ell := \prod_{i=k}^{\ell}(1-p'_k)^{i-k}p'_k - \prod_{i=k}^{\ell}(1-p_k)^{i-k}p_k$, which implies that $\Gamma_\ell = (1-p_k)^{\ell-k+1} - (1-p'_k)^{\ell-k+1} \geq 0$ for any $\ell = k,\dots,j$, since $p_k \leq p'_k$. Similar with (20), we have

$$-\Gamma_{\ell+1} + p'_k(1-p'_k)^{\ell-k+1} = -\Gamma_\ell + p_k(1-p_k)^{\ell-k+1}. \tag{21}$$

Then, we can get that

$$\begin{aligned}\bar{\mu}(p_k,\dots,p_j) &= \sum_{\ell=k}^{j}(1-p_k)^{\ell-k}p_k\cdot\ \ 1+\bar{\mu}(p_{\ell+1},\dots,p_j)\\ &= -\Gamma_k\ \ 1+\bar{\mu}(p_{k+1},\dots,p_j)\ \ + p'_k\ \ 1+\bar{\mu}(p_{k+1},\dots,p_j)\ \ + \sum_{\ell=k+1}^{j}(1-p_k)^{\ell-k}p_k\cdot\ \ 1+\bar{\mu}(p_{\ell+1},\dots,p_j)\\ &\leq -\Gamma_{k+1}\ \ 1+\bar{\mu}(p_{k+2},\dots,p_j)\ \ + \sum_{\ell=k}^{k+1}(1-p'_k)^{\ell-k}p'_k\cdot\ \ 1+\bar{\mu}(p_{\ell+1},\dots,p_j)\\ &\quad + \sum_{\ell=k+2}^{j}(1-p_k)^{\ell-k}p_k\cdot\ \ 1+\bar{\mu}(p_{\ell+1},\dots,p_j)\\ &\leq \cdots \leq -\Gamma_j + \bar{\mu}(p'_k,\dots,p_j) \leq \bar{\mu}(p'_k,\dots,p_j),\end{aligned}$$

where the inequality is due to Lemma 20 and (21). Thus, Lemma 21 follows.

Given any $\beta = 2,\dots,n$, we divide V into $L(\beta)+1$ buckets such that bucket $B_k = \{x_i : \beta^k \leq i < \beta^{k+1}\}$ with $k = 0,1,\dots,L(\beta)$, where $L(\beta) = \lfloor \log_\beta n \rfloor$. For $x_i \in B_k$, let $\hat{p}_i := p_{\beta^k}$ which is an upper bound on p_i. The following lemma establishes an upper bound on $\bar{\mu}$.

Lemma 22. *Given $\beta = 2,\dots,n$, $\bar{\mu} \leq \hat{\mu}(\beta) + L(\beta) + 1$, where $\hat{\mu}(\beta) = \prod_{i=1}^{n}\hat{p}_i$.*

Proof. Define $\bar{\mu}^* := \bar{\mu}(\hat{p}_1, \ldots, \hat{p}_n)$. By Lemma 21, we have $\bar{\mu} \le \bar{\mu}^*$. We just need to show that $\bar{\mu}^* \le \hat{\mu}(\beta) + L(\beta) + 1$. By Lemma 20, we have

$$\begin{aligned}
\bar{\mu}^* &= \sum_{k=1}^{n} (1-\hat{p}_1)^{k-1} \cdot (1 + \bar{\mu}(\hat{p}_{k+1}, \ldots, \hat{p}_n)) \\
&\le \sum_{k=1}^{\beta-1} (1-\hat{p}_1)^{k-1}\hat{p}_1 \cdot (1 + \bar{\mu}(\hat{p}_{k+1}, \ldots, \hat{p}_n)) \;+ \sum_{k=\beta}^{n} (1-\hat{p}_1)^{k-1}\hat{p}_1 \cdot \; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \\
&\le \sum_{k=1}^{\beta-1} (1-\hat{p}_1)^{k-1}\hat{p}_1 \cdot (1 + \bar{\mu}(\hat{p}_{k+1}, \ldots, \hat{p}_n)) \;+(1-\hat{p}_1)^{\beta-1} \cdot \; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \;\; .
\end{aligned}$$

For $\beta = 2$, it is straightforward to verify that,

$$\begin{aligned}
\mu^* &\le \hat{p}_1 \;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \;\; + (1-\hat{p}_1) \;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \\
&= 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \le \hat{p}_1 + 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n).
\end{aligned}$$

For $\beta > 2$, we have

$$\bar{\mu}(\hat{p}_2, \ldots, \hat{p}_n) \le \sum_{k=2}^{\beta-1} (1-\hat{p}_2)^{k-2}\hat{p}_2 \cdot (1 + \bar{\mu}(\hat{p}_{k+1}, \ldots, \hat{p}_n)) \;\; +(1-\hat{p}_2)^{\beta-2} \cdot \;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \;\; .$$

Note that $\hat{p}_1 = \hat{p}_2$ when $\beta > 2$. Thus, $(1-\hat{p}_1)^{k-1}\hat{p}_1 + \hat{p}_1(1-\hat{p}_2)^{k-2}\hat{p}_2 = (1-\hat{p}_1)^{k-2}\hat{p}_1$. As a result,

$$\begin{aligned}
\bar{\mu}^* &\le \hat{p}_1 + \sum_{k=2}^{\beta-1} \hat{p}_1(1-\hat{p}_2)^{k-2}\hat{p}_2 + (1-\hat{p}_1)^{k-1}\hat{p}_1) \;\; (1 + \bar{\mu}(\hat{p}_{k+1}, \ldots, \hat{p}_n)) \\
&\quad + \;\; \hat{p}_1(1-p_2)^{\beta-2} + (1-\hat{p}_1)^{\beta-1} \;\;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \\
&\le \hat{p}_1 + \sum_{k=2}^{\beta-1} (1-\hat{p}_1)^{k-2}\hat{p}_1 \cdot (1 + \bar{\mu}(\hat{p}_{k+1}, \ldots, \hat{p}_n)) \;\; +(1-\hat{p}_1)^{\beta-2} \cdot \;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \\
&\le \cdots \le (\beta-2)\hat{p}_1 + \hat{p}_1 \cdot \;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \;\; + (1-\hat{p}_1) \cdot \;\; 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \\
&\le (\beta-1)\hat{p}_1 + 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n).
\end{aligned}$$

Therefore for $\beta = 2, \ldots, n$, we recursively have

$$\begin{aligned}
\bar{\mu}^* &\le (\beta-1)\hat{p}_1 + 1 + \bar{\mu}(\hat{p}_\beta, \ldots, \hat{p}_n) \le (\beta-1)\hat{p}_1 + (\beta^2-\beta)\hat{p}_\beta + 2 + \bar{\mu}(\hat{p}_{\beta^2}, \ldots, \hat{p}_n) \\
&\le \cdots \le \sum_{k=0}^{L(\beta)-1} (\beta^{k+1} - \beta^k)\hat{p}_{\beta^k} + L(\beta) + \bar{\mu}(\hat{p}_{\beta^{L(\beta)}}, \ldots, \hat{p}_n).
\end{aligned}$$

Meanwhile, SKIP performs sampling with standard geometric distribution $G(\hat{p}_{\beta^{L(\beta)}})$ on $\{\hat{p}_{\beta^{L(\beta)}}, \ldots, \hat{p}_n\}$, which indicates that $\bar{\mu}(\hat{p}_{\beta^{L(\beta)}}, \ldots, \hat{p}_n) = (n - \beta^{L(\beta)} + 1)\hat{p}_{\beta^{L(\beta)}} + 1$. Therefore, $\bar{\mu}^* \le \hat{\mu}(\beta) + L(\beta) + 1$.

Now, we are ready to prove our main result.

Proof of Theorem 4. Recall that SKIP takes an expected time of $O(1+\bar{\mu})$. By Lemma 22, we know that $\bar{\mu} \le \min_{\beta \in \{2,\ldots,n\}}\{\hat{\mu}(\beta) + L(\beta) + 1\}$. In addition, according to the definition of $\hat{p}_i$ under a given β, it is easy to verify that $\hat{p}_i \le p_{\lceil i/\beta \rceil}$. Thus,

$$\hat{\mu}(\beta) = \sum_{i=1}^{n} \hat{p}_i \le \sum_{i=1}^{n} p_{\lceil i/\beta \rceil} \le \beta \sum_{i=1}^{n} p_i = \beta\mu.$$

Therefore, $\bar{\mu} \leq \min_{\beta \in \{2,\ldots,n\}} \{\beta\mu + \log_\beta n + 1\}$.

When $\mu \geq (\log n)/2$, $\bar{\mu} \leq 2\mu + \log_2 n \leq 6\mu$ by setting $\beta = 2$. Thus, Theorem 4 holds, since $O(1+\bar{\mu}) = O(\mu)$.

Next, we consider $\mu \ddot{Y} (\log n)/2$. Define

$$\gamma := \frac{(\log n)/\mu}{\log((\log n)/\mu)} \quad \text{and} \quad \beta^* := \lceil \gamma \rceil.$$

Thus, $\beta^*\mu = O(\frac{\log n}{\log((\log n)/\mu)})$ and $\log_{\beta^*} n = O(\frac{\log n}{\log((\log n)/\mu)})$. Therefore,

$$O(1+\bar{\mu}) = O\left(1 + \frac{\log n}{\log((\log n)/\mu)}\right).$$

This completes the proof.

B VARIANT OF GREEDY ALGORITHM

In this appendix, we present another revised greedy algorithm, which is different from Greedy-Degree in Alg. 9. For ease of explanation, we name it *Greedy-Cost*.

Recap that in the second phase of HIST, we can stop the RR set generation process as soon as we hit any sentinel node, thus reducing the average size of the RR sets. Suppose we have a function $C_{\mathcal{R}}(S)$ which represents the amount of cost reduction on the collection $\mathcal{R}$ of RR sets when S is selected as the sentinel set. Then we can design the Greedy-Cost algorithm by replacing Line 3 in Alg. 1, that is,

$$v \leftarrow \arg\max_{u \in V} (\Lambda_{\mathcal{R}}(S_k^* \cup \{u\})) - \Lambda_{\mathcal{R}}(S_k^*)$$

with the following statements:

$$\mathcal{M} \leftarrow \arg\max_{u \in V} (\Lambda_{\mathcal{R}}(S_k^* \cup \{u\})) - \Lambda_{\mathcal{R}}(S_k^*)$$
$$v \leftarrow \arg\max_{v' \in \mathcal{M}} C_{\mathcal{R}}(S_k^* \cup \{v'\}) - C_{\mathcal{R}}(S_k^*)$$

where $\mathcal{M}$ is the set of nodes which have the largest marginal coverage. Obviously if $|\mathcal{M}| = 1$, we have only one candidate, and it must be selected as the sentinel node.

In the following, we present how to define the cost funtion $C_{\mathcal{R}}(\cdot)$. Recap that when generating an RR set R, each sampled node is added into R one by one (please see Alg. 2). Thus, we say the index of u is i if it is the i-th node added into R, where $i = 0, 1, 2, \ldots, |R| - 1$. Let $l(u, R)$ be the function which returns the index of u in R. Let $l(u, R) = |R|$ if $u \not\in R$. Due to the existence of the sentinel set S, the process of generating R can be stopped immediately when it reaches the sentinel node $u^* \in S$ with the minimum index,

$$u^* = \arg\min_{u' \in S} l(u', R).$$

Therefore, we define the cost reduction function on an RR set R caused by the sentinel set S as:

$$C(R, S) = |R| - l(u^*, R).$$

It is easy to realize that if no node of S is hit by R, the cost reduction $C(R, S) = 0$ due to $l(u, R) = |R|$ for each $u \in S$. It implies that it can not save any sampling cost when generating R. By summing up all the cost reduction in $\mathcal{R}$, the function $C_{\mathcal{R}}(S)$ is defined as

$$C_{\mathcal{R}}(S) = \sum_{R \in \mathcal{R}} C(R, S).$$