Influence Maximization Revisited: Efficient Reverse Reachable Set Generation with Bound Tightened

Qintian Guo
The Chinese University of Hong Kong
qtguo@se.cuhk.edu.hk

Sibo Wang*
The Chinese University of Hong Kong
swang@se.cuhk.edu.hk

Zhewei Wei
Renmin University of China
zhewei@ruc.edu.cn

Ming Chen
Renmin University of China
chennnming@ruc.edu.cn

ABSTRACT

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. 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.

*Sibo Wang and Zhewei Wei are the corresponding authors.

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3389740

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**.

KEYWORDS

Influence Maximization; Sampling

ACM Reference Format:
Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. 2020. Influence Maximization Revisited: Efficient Reverse Reachable Set Generation with Bound Tightened. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3318464.3389740

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. [26] 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. [8] 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 [34, 37–39]. Tang et al. [39] 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. [38] propose *IMM*, Nguyen et al. [34] develop *SSA* and *D-SSA*, and Tang et al. propose *OPIM-C* [37] 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. 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.

[1]The lower bound in [8] 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 $m=\|E\|$
$d_{in}(v)$	the in-degree of node v
$\mathbb{I}_{\mathbb{C}}(S)$	the expected influence of S
OPT_k	the maximal $\mathbb{I}_{\mathbb{C}}(S)$ for any size-k seed set
S^o_k	an optimal seed set with $\mathbb{I}_{\mathbb{C}}(S^o_k)=OPT_k$
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 $\mathcal{R}$
$\mathbb{I}^-_{\mathbb{C}}(S)$	a lower bound of the expected influence of S
$\mathbb{I}^+_{\mathbb{C}}(S^o_k)$	an upper bound of the expected influence of S^o_k

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 $\mathbb{C}$ 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 $\mathbb{C}$. We denote set S as the seed set and $I_C(S)$ as the influence of S in stochastic propagation instance C, and denote $\mathbb{I}_{\mathbb{C}}(S) = \mathbb{E}_{C \in \mathbb{C}}[I_C(S)]$ as the expected influence of S under the cascade process $\mathbb{C}$. Table 1 lists the notations used frequently in this paper.

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

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. [8]. 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.

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

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

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

v. Borgs et al. [8] 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 $\mathbb{C}$, then*

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

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 $S \cap R \neq \emptyset$. 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. [8] 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 *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 select 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. [39] 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. [38] 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.

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

```
1 Randomly sample a node $v \in V$ 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     for each in-neighbor $w$ of $u$ do
6         if $w$ is inactivated and $rand() \le p(w,u)$ then
7             Add $w$ to $R$;
8             Add $w$ queue $Q$ and mark $w$ as activated;
9     return $R$;
```

Lemma 2 ([38]). *Given a fixed number θ of random RR sets and a seed set S, for any $\lambda > 0$,*

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

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

As shown in [38], *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. [34] 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. [24] 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 *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. [33] 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 [24, 37].

OPIM-C. The latest RR set based solution for IM is the *OPIM-C* algorithm [37]. *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 $\mathbb{I}_{\mathbb{C}}^+(S_k^o)$ of $\mathbb{I}_{\mathbb{C}}(S_k^o)$. Next, they sample another set $\mathcal{R}_2$ of random RR sets with $|\mathcal{R}_2| = |\mathcal{R}_1|$ and derive a lower bound $\mathbb{I}_{\mathbb{C}}^-(S_k^*)$ of $\mathbb{I}_{\mathbb{C}}(S_k^*)$. The algorithm terminates as soon as

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

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 $\mathbb{I}_{\mathbb{C}}^+(S_k^o)$ of $\mathbb{I}_{\mathbb{C}}(S_k^o)$. With tighter bounds, the number of RR set samples can be reduced, thus improving the running time. By applying Lemma 2, Tang et al.[37] derive the lower bound $\mathbb{I}_{\mathbb{C}}^-(S_k^*)$ as follows:

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

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 $\mathbb{I}_{\mathbb{C}}^+(S_k^o)$ is given as follows:

$$\mathbb{I}_{\mathbb{C}}^+(S_k^o) = \left(\sqrt{\Lambda_{\mathcal{R}_1}^u(S_k^o) + \frac{\eta_u}{2}} + \sqrt{\frac{\eta_u}{2}}\right)^2\cdot\frac{n}{\theta_1}, \tag{2}$$

where $\eta_u = \ln(1/\delta_u)$ and δ_u is the probability that the above upper bound fails; $\Lambda_{\mathcal{R}_1}^u(S_k^o)$, an upper bound of the coverage of S_k^o with respect to $\mathcal{R}_1$, is derived as follows. Though the optimal seed set S_k^o is unknown, the upper bound $\Lambda_{\mathcal{R}_1}^u(S_k^o)$ 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,

$$\Lambda_{\mathcal{R}_1}^u(S_k^o) = \min_{0\le i\le k}\left(\Lambda_{\mathcal{R}_1}(S_i^*) + \sum_{v\in maxMC(S_i^*,k)}\Lambda_{\mathcal{R}_1}(v|S_i^*)\right).$$

2.3 RR Set Generation

All of the above solutions 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 direction 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 [37] on how to generate a random RR set under LT model.

According to [8, 39], a random RR set can be constructed in $O(\frac{m}{n}\cdot\mathbb{I}_{\mathbb{C}}(v^*))$ expected time, where $\mathbb{I}_{\mathbb{C}}(v^*)$ is the expected influence of a node v^* sampled from V where each v is sampled with a probability of $d_{in}(v)/m$. However, no further research has presented any theoretical study on this RR set generation phase. We next present a study to fill this gap.

Algorithm 3: SUBSIM(G)

```
1  Randomly sample a node $v \in V$ 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 $u[i]$ ($i = 1, 2, \ldots$) be the $i$th in-neighbor of $u$;
6      $p \leftarrow \frac{1}{d_{in}(u)}$ under WC;
7      $i \leftarrow \lceil \log(rand())/\log(1-p) \rceil$;
8      while $i \leq d_{in}(u)$ do
9          $w \leftarrow u[i]$;
10         if $w$ is not activated then
11             Add $w$ to $R$;
12             Add $w$ to queue $Q$ and mark $w$ as activated;
13         $i$ += $\lceil \log(rand())/\log(1-p) \rceil$;
14     return $R$;
```

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 3.3.

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 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 algorithms such 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 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 $i > h$, it indicates that no element is sampled from set S. Notice that in distribution 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 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 [27] to sample from distribution G(p): Given a U generated uniformly at random from $(0, 1)$, we can sample h' from 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 G(p). Therefore, the expected cost of the sampling phase only depends on the number of times we do geometric sampling and we have the following lemma.

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 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 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, we have 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 $\theta(m/n)\cdot \mathbb{I}_{\mathbb{C}}(\{v^*\})$, 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 u is activated by v, and with the fact that the expected cost to examine (w,u) is $p(w,u)$ under geometric sampling, we can derive that:

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

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

$$\mathbb{E}[R(v)] = \sum_{(w,u)\in E} \Pr[v \xrightarrow{R} (w,u)] = \sum_{(w,u)\in E} \Pr[v \xrightarrow{R} u]\cdot p(w,u)$$

$$= \sum_{u\in V} \Pr[v \xrightarrow{R} u]\cdot \sum_{(w,u)\in E} p(w,u) \le \sum_{u\in V} \theta(d_{in}(u))\cdot \Pr[v \xrightarrow{R} u]$$

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

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

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:

$$E_R \le \frac{\theta(V)}{n}\sum_{v\in V}\sum_{u\in V}\frac{\theta(d_{in}(u))}{\theta(V)}\cdot \Pr[u\to v]$$
$$= \frac{\theta(V)}{n}\sum_{u\in V}\frac{\theta(d_{in}(u))}{\theta(V)}\sum_{v\in V}\Pr[u\to v]$$

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:

$$E_R \le \frac{\theta(V)}{n}\sum_{u\in V}\frac{\theta(d_{in}(u))}{\theta(V)}\mathbb{I}_{\mathbb{C}}(\{u\})$$
$$= \frac{\theta(V)}{n}\cdot \mathbb{I}_{\mathbb{C}}(\{v^*\}) \le \theta(m/n)\cdot \mathbb{I}_{\mathbb{C}}(\{v^*\})$$

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, $\sum_{(u,v)\in E} p(u,v) \le \theta(d_{in}(v))$, 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 [38] 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 $\mathbb{I}_{\mathbb{C}}(\{v^*\}) \le OPT_k$, we know that $E_R = O(\theta(m/n)\cdot OPT_k)$. Combining them together, we derive the time complexity:

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

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 [37, 38], and it naturally holds that $\sum_{u\in IN(v)} p(u,v) \le 1$, 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 easily derived that the time complexity of existing IM algorithms under LT model can be reduced to $O(k\cdot n\cdot\log n\cdot\epsilon^{-2})$.

3.3 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 [38], or by learning from data [19, 20]. In this section, we discuss how to handle general IC model. We still map the selection of in-neighbors to subset sampling and have the following lemma from [9] 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 apply geometric sampling to sample a position h'. When $h' \le |B_k|$, 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'-the 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 ($i<j$) 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 [41]. 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 $O(1+\mu)$. 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, it requires complicated preprocessing indices, which may hamper the practical performance on sparse graphs. To tackle this issue, we present an index-free solution that runs in $O(k \cdot \log(m/n) \cdot n \cdot \log n/\epsilon^2)$ expected time with existing IM algorithms if $\sum_{(u,v)\in E} p(u,v) = O(\log(d_{in}(v)))$ for any node v. The solution only requires the incoming edges of the same node to be sorted in descending order of their weights.

Index-free method. According to [9], if the elements $x_1, x_2, \cdots, x_h$ of set S are sorted in descending order of their probability ($p_1, p_2, \cdots, p_h$, respectively), one can do subset sampling as follows to achieve $O(1+\mu+\log h)$ expected sampling cost. In particular, we do bucketing by their sorted positions such that elements whose positions fall into the range $[2^k, 2^{k+1})$ belong to bucket B_k. Then, for bucket B_k, we use p_{2^k} as the probability for geometric distribution $G(p_{2^k})$. When h' is sampled from $G(p_{2^k})$, if $2^k+h' \geq 2^{k+1}$, no element is sampled from bucket B_k. Otherwise, it skips $h'-1$ elements and jumps to position $2^k+h'$. To guarantee that the probability to sample $x_{2^k+h'}$ is still $p_{2^k+h'}$, we further sample a random number U and only sample element $x_{2^k+h'}$ if U is no more than $p_{2^k+h'}/p_{2^k}$. Since $p_x \leq p_{\lceil x/2 \rceil}$, we can bound the total expected cost to sample from each bucket to $O(1+\mu+\log h)$, where the $\log h$ term comes from the number of buckets. The above strategy can be easily implemented without additional indices. When $\sum_{(u,v)\in E} p(u,v) = O(\log(d_{in}(v)))$, the total cost to sample the in-neighbors of node v can be bounded by $O(\log(d_{in}(v)))$. We can immediately apply Theorem 1 and obtain that existing IM algorithms can achieve a time complexity of $O(k \cdot \log(m/n) \cdot n \cdot \log n/\epsilon^2)$ when using the above sampling strategy in RR set generation.

Algorithm 4: HIST(G, k, ϵ, δ)

```
1 $\epsilon_1 = \epsilon_2 = \epsilon/2$, $\delta_1 = \delta_2 = \delta/2$;
2 $S^*_b$ = SentinelSet($G, k, \epsilon_1, \delta_1$);
3 $S^*_k$ = IM-Sentinel($G, k, \epsilon, S^*_b, \epsilon_2, \delta_2$) ;
4 return $S^*_k$;
```

4 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 is the main cause of 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 is likely to 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 of HIST, and with the help of S^*_b, subsequent RR sets can be generated efficiently in the second phase of HIST 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 5).

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 $\mathbb{I}_{\mathbb{C}}(S^*_b) \geq (1-(1-1/k)^b-\epsilon_1) \cdot \mathbb{I}_{\mathbb{C}}(S^o_k)$ with high probability, where S^o_k is the optimal seed set.
- 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

Algorithm 5: 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.
```

Algorithm 6: Revised-Greedy($G, \mathcal{R}, k$)

```
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.
```

$b = k$, the sentinel set selection phase falls into existing IM algorithms that provides a $(1-(1-1/k)^k - \epsilon)$ ($\approx 1 - 1/e - \epsilon$) approximate solution. When $b < k$, 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 6). 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 pseudo-code of the HIST algorithm is shown in Algorithm 4, 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.

4.1 Sentinel Set Selection Phase

Algorithm 7 shows the pseudo-code 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 7 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 7 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 6) so as to better achieve the goal. When two 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 6 compared to Algorithm 1 which regards all nodes with the same importance as long as their marginal coverage is the same.

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 6 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 $\mathbb{I}_{\mathbb{C}}^{+}(S_k^o)$ for $\mathbb{I}_{\mathbb{C}}(S_k^o)$. However, we can not use $\mathcal{R}_1$ to derive a lower bound of $\mathbb{I}_{\mathbb{C}}(S_a^*)$. 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

Algorithm 7: SentinelSet$(G, k, \epsilon_1, \delta_1)$

```
1  Set $\theta_0 = 3 \cdot \ln(1/\delta_1)$ and $\theta_{max}$ according to Eqn. 3;
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 6 with $\mathcal{R}_1$ as the input;
6      Estimate the lower bound $\hat{\mathbb{I}}_{\mathbb{C}}^-(S_a^*)$ based on the
         result of Line 5, where $a \in \{1 \ldots k\}$;
7      Compute $\mathbb{I}_{\mathbb{C}}^+(S_k^o)$ by Eqn. 2, setting $\delta_u = \frac{\delta_1}{3 i_{max}}$;
8      Let $b$ be the maximum $a$ such that
         $\hat{\mathbb{I}}_{\mathbb{C}}^-(S_a^*)/\mathbb{I}_{\mathbb{C}}^+(S_k^o) > (1 - (1 - \frac{1}{k})^a - \epsilon_1)$;
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 $\mathbb{I}_{\mathbb{C}}^-(S_b^*)$ by Eqn. 1; set $\delta_l = \frac{\delta_1}{6 i_{max}}$;
11     if $\mathbb{I}_{\mathbb{C}}^-(S_b^*)/\mathbb{I}_{\mathbb{C}}^+(S_k^o) > (1 - (1 - \frac{1}{k})^b - \epsilon_1)$ then
12         return $S_b^*$;
13     Increase the size of $\mathcal{R}_2$ to $4|\mathcal{R}_1|$ and compute
         $\mathbb{I}_{\mathbb{C}}^-(S_b^*)$ again;
14     if $\mathbb{I}_{\mathbb{C}}^-(S_b^*)/\mathbb{I}_{\mathbb{C}}^+(S_k^o) > (1 - (1 - \frac{1}{k})^b - \epsilon_1)$ then
15         return $S_b^*$;
16     double the size of $\mathcal{R}_1$;
17 return $S_b^*$;
```

to derive an estimation of the lower bound on $\mathbb{I}_{\mathbb{C}}(S_a^*)$, denoted as $\hat{\mathbb{I}}_{\mathbb{C}}^-(S_a^*)$, as if $\mathcal{R}_1$ and S_a^* were independent. Then, we select the maximum a (Algorithm 7 Line 8) such that:

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

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 $\mathbb{I}_{\mathbb{C}}^-(S_b^*)$ using Equation 1. Given $\mathbb{I}_{\mathbb{C}}^-(S_b^*)$, we are able to check if S_b^* satisfies the approximation ratio (Algorithm 7 Line 11), i.e.,

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

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 $\mathbb{I}_{\mathbb{C}}^-(S_b^*)$ for S_b^* (Algorithm 7 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 7 Line 16) and repeat the whole process until we find the seed set S_b^* satisfying the approximation ratio.

Stopping condition. We now give the following lemma to establish the stopping condition of Algorithm 7. It provides a bound on the number of random RR sets required in $\mathcal{R}_1$ in the sentinel set selection phase.

Lemma 6. *Let $\mathcal{R}_1$ be the set of random RR sets generated by Algorithm 7 and S_b^* be a size-b node set selected by Algorithm 6 on $\mathcal{R}_1$. Given ϵ' and δ', if the size of $\mathcal{R}_1$ satisfies*

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

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

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

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 6, by replacing $\mathbb{I}_{\mathbb{C}}(S_k^o)$ 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:

$$\theta_{max} = \frac{2n\left(\sqrt{\ln\frac{6}{\delta_1}} + \sqrt{(\ln\binom{n}{k} + \ln\frac{6}{\delta_1})}\right)^2}{\epsilon_1^2 \cdot k}. \quad (3)$$

That is, if the size of the set $\mathcal{R}_1$ is θ_{max}, the seed set S_b^* selected by Algorithm 6 guarantees $(1 - (1 - 1/k)^b - \epsilon_1)$ approximation of $\mathbb{I}_{\mathbb{C}}(S_k^o)$ 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 $3\ln(1/\delta)/\mu/\epsilon^2$ [16] 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 7 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 7 ensures $(1 - (1 - 1/k)^b - \epsilon_1)$ approximation with at least $1 - \delta_1$. The algorithm has at most i_{max} iterations. In the last iteration, no matter whether $\mathbb{I}_{\mathbb{C}}^-(S_b^*)/\mathbb{I}_{\mathbb{C}}^+(S_k^o)$ reaches the approximation threshold or not, it returns S_b^* as the final seed set. As shown in Lemma 6, θ_{max} RR samples ensure that the failure probability of S_b^* being unqualified, i.e. $\mathbb{I}_{\mathbb{C}}(S_b^*) < (1 - (1 - 1/k)^b - \epsilon_1)\mathbb{I}_{\mathbb{C}}(S_k^o)$, 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 7 is at most δ_1.

4.2 IM-Sentinel Phase

Algorithm 8 shows the pseudo-code of the IM-Sentinel phase. In this phase, we apply Algorithm 5 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 8 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 4 (Algorithm 8 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 6 and derives the upper bound $\mathbb{I}_{\mathbb{C}}^+(S_k^o)$ (Algorithm 8 Lines 5-8), and uses the other set $\mathcal{R}_2$ to derive the lower bound $\mathbb{I}_{\mathbb{C}}^-(S_k^*)$ (Algorithm 8 Line 9). When the approximation ratio is satisfied, we return the seed set S_k^* (Algorithm 8 Line 10-11). Otherwise, we double the size of $\mathcal{R}_1$ and $\mathcal{R}_2$ and repeat the above process.

The main difference is that, when generating $\mathcal{R}_1$ and $\mathcal{R}_2$, we can apply Algorithm 5 to effectively reduce the size of a random RR set. Besides, when we feed $\mathcal{R}_1$ to Algorithm 6 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 8 Line 5).

Stopping condition. Here we provide another lemma to bound the size of $\mathcal{R}_1$ in Algorithm 8.

Lemma 7. *Let S_b^* be the seed set returned by Algorithm 7. Given ϵ' and δ', if the number of the set $\mathcal{R}_1$ satisfies*

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

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

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

According to Lemma 7, we replace $\mathbb{I}_{\mathbb{C}}(S_k^o)$ with k, set $\delta' = \delta_2/3, \epsilon' = \epsilon_2$, and define the maximum number of RR sets in the IM-Sentinel phase as

$$\theta_{max} = \frac{2n \cdot \left(\sqrt{\ln\frac{9}{\delta_2}} + \sqrt{(1-1/e)(\ln\binom{n-b}{k-b} + \ln\frac{9}{\delta_2})}\right)^2}{\epsilon_2^2 \cdot k}. \quad (4)$$

That is, if the size of $\mathcal{R}_1$ is θ_{max}, the seed set S_{k-b}^* obtained in Algorithm 8 Line 6 guarantees $\mathbb{I}_{\mathbb{C}}(S_b^* \cup S_{k-b}^*) \geq (1 - 1/e - \epsilon_1 - \epsilon_2)\mathbb{I}_{\mathbb{C}}(S_k^o)$ with at least $1 - \delta_2/3$ probability.

Algorithm 8: IM-Sentinel($G, k, \epsilon, S_b^*, \epsilon_2, \delta_2$)

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

Failure probability. Like the analysis of Algorithm 7, 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 8 is δ_2.

5 THEORETICAL ANALYSIS

In this section, we provide the proofs of Lemma 6 and 7.

Proof of Lemma 6. We first give three lemmas that will be used in the proof of Lemma 6.

Lemma 8. *Let S_b^* be the seed set selected by Algorithm 6. Let $x = (1 - 1/k)$, then $\Lambda_{\mathcal{R}}(S_b^*) \geq (1 - x^b)\Lambda_{\mathcal{R}}(S_k^o)$.*

Proof. We denote $S_j^*(1 \leq j \leq b)$ as the set of nodes selected in the first j iterations of the modified greedy algorithm and let $S_0^* = \emptyset$. Let $\Lambda_{\mathcal{R}}(v|S)$ be the marginal coverage of a node v in $\mathcal{R}$ with respect to a seed set S. Using the submodularity of coverage function $\Lambda_{\mathcal{R}}(\cdot)$,

$$\begin{aligned}\Lambda_{\mathcal{R}}(S_k^o) \leq \Lambda_{\mathcal{R}}(S_j^* \cup S_k^o) &\leq \Lambda_{\mathcal{R}}(S_j^*) + \sum_{v \in S_k^o \setminus S_j^*} \Lambda_{\mathcal{R}}(v|S_j^*) \\ &\leq \Lambda_{\mathcal{R}}(S_j^*) + k\left(\Lambda_{\mathcal{R}}(S_{j+1}^*) - \Lambda_{\mathcal{R}}(S_j^*)\right).\end{aligned}$$

Define $\gamma_j = \Lambda_{\mathcal{R}}(S_k^o) - \Lambda_{\mathcal{R}}(S_j^*)$. We have $\gamma_{j+1} \leq \left(1 - \frac{1}{k}\right)\gamma_j$ Recursively, we have that:

$$\Lambda_{\mathcal{R}}(S_k^o) - \Lambda_{\mathcal{R}}(S_b^*) \leq x^b\left(\Lambda_{\mathcal{R}}(S_k^o) - \Lambda_{\mathcal{R}}(S_0^*)\right) = x^b\Lambda_{\mathcal{R}}(S_k^o).$$

It then can be rearranged as $\Lambda_{\mathcal{R}}(S_b^*) \geq (1 - x^b)\Lambda_{\mathcal{R}}(S_k^o)$. □

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

Lemma 9. *Given δ_1', ϵ_1', and $\theta_1 = \frac{2n\cdot\ln(1/\delta_1')}{\mathbb{I}_{\mathbb{C}}(S_k^o)\cdot\epsilon_1'^2}$, if $\theta \geq \theta_1$, $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o) \geq (1-\epsilon_1')\mathbb{I}_{\mathbb{C}}(S_k^o)$ holds with $1-\delta_1'$ probability.*

Refer to [6] for the proof of Lemma 9. If θ is large, $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*)$ is close to $\mathbb{I}_{\mathbb{C}}(S_b^*)$. Based on Lemmas 8-9, we have:

$$\frac{n}{\theta}\cdot\Lambda_{\mathcal{R}}(S_b^*) \geq \left(1-x^b\right)(1-\epsilon_1')\mathbb{I}_{\mathbb{C}}(S_k^o). \tag{5}$$

Hence, it is possible for us to build a connection between $\mathbb{I}_{\mathbb{C}}(S_b^*)$ and $\mathbb{I}_{\mathbb{C}}(S_k^o)$, which is the following lemma.

Lemma 10. *Given $\delta_2', \epsilon_1' < \epsilon'$, if Equation 5 holds and*

$$\theta > \theta_2 = \frac{2(1-x^b)\cdot n\cdot\left(\ln\binom{n}{b}+\ln\frac{1}{\delta_2'}\right)}{\mathbb{I}_{\mathbb{C}}(S_k^o)\cdot\left(\epsilon'-(1-x^b)\cdot\epsilon_1'\right)^2},$$

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

Refer to [6] for the proof of Lemma 10.

Now we give the proof of Lemma 6. Based on Lemma 9 and Lemma 10 and by the union bound, if $\theta > \max(\theta_1, \theta_2)$, it holds that $\mathbb{I}_{\mathbb{C}}(S_b^*) \geq (1-x^b-\epsilon')\mathbb{I}_{\mathbb{C}}(S_k^o)$ 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

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

Hence, if $\theta > \theta'$, S_b^* satisfies $\mathbb{I}_{\mathbb{C}}(S_b^*) \geq (1-x^b-\epsilon')\mathbb{I}_{\mathbb{C}}(S_k^o)$ with at least $1-\delta'$ probability.

Proof of Lemma 7. We first give several lemmas that will be used in the proof of Lemma 7.

Lemma 11. *Let S_b^* be the seed set returned by Algorithm 7. Let S_{k-b}^* be the seed set generated in Algorithm 8 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 $S_j^*(1\leq j\leq k-b)$ be the set of nodes selected in the first j iterations of the generation of S_{k-b}^*, and $M_j(1\leq j\leq k-b)$ 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 $\Lambda_{\mathcal{R}}(\cdot)$,

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

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

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

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

The lemma is proved. □

According to Lemma 9, given δ_1', ϵ_1', and $\theta_1 = \frac{2n\ln\frac{1}{\delta_1'}}{\mathbb{I}_{\mathbb{C}}(S_k^o)\cdot\epsilon_1'^2}$, if the size of $\mathcal{R}$, denoted as θ, is larger than θ_1, it follows that $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o) \geq (1-\epsilon_1')\mathbb{I}_{\mathbb{C}}(S_k^o)$ with at least $1-\delta_1'$ probability. In fact, at this moment ($\theta > \theta_1$), $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*)$ is close to $\mathbb{I}_{\mathbb{C}}(S_b^*)$. That is the following lemma.

Lemma 12. *Given ϵ_1' and δ_1', if $\theta > \theta_1$, it holds that $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*) \geq \mathbb{I}_{\mathbb{C}}(S_b^*)-\epsilon_1'\mathbb{I}_{\mathbb{C}}(S_k^o)$ 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 $p = \mathbb{I}_{\mathbb{C}}(S_b^*)/n$. Obviously, $\mathbb{I}_{\mathbb{C}}(S_k^o) > \mathbb{I}_{\mathbb{C}}(S_b^*) = np$. We have

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

The lemma follows. □

Combining Lemma 11 and 12, we have that:

Lemma 13. *Given δ_1', ϵ_1', if $\theta > \theta_1$ and $\mathbb{I}_{\mathbb{C}}(S_b^*) \geq (1-x^b-\epsilon_1)\mathbb{I}_{\mathbb{C}}(S_k^o)$, then with at least $1-2\delta_1'$, we have*

$$\frac{n}{\theta}\cdot\Lambda_{\mathcal{R}}(S_b^*\cup S_{k-b}^*) \geq (1-1/e-\epsilon_1-\epsilon_1')\mathbb{I}_{\mathbb{C}}(S_k^o). \tag{6}$$

Proof. Based on Lemma 11,

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

When $\theta > \theta_1$, both $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_k^o) \geq (1-\epsilon_1')\mathbb{I}_{\mathbb{C}}(S_k^o)$ and $\frac{n}{\theta}\Lambda_{\mathcal{R}}(S_b^*) \geq \mathbb{I}_{\mathbb{C}}(S_b^*)-\epsilon_1'\mathbb{I}_{\mathbb{C}}(S_k^o)$ 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 14. *Given $\delta_2', \epsilon_1' < \epsilon'$, and*

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

if Equation 6 holds and $\theta > \theta_2$, then

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

Refer to [6] for the proof of Lemma 14.

Now we prove Lemma 7. Lemmas 13 and 14 hold with $1-2\delta_1'$ and $1-\delta_2'$ probability, respectively. By union bound, if $\theta > \max(\theta_1, \theta_2)$, with $1-2\delta_1'-\delta_2'$ probability, we have that:

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

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

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

The lemma follows.

6 ADDITIONAL RELATED WORK

There has been a large body of research on IM, e.g., [11–15, 17, 20–22, 25, 26, 28, 29, 31, 35, 42], in the literature. Kempe et al. [26] 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, 12–14, 17, 21, 22, 25, 35], 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 [8, 34, 37–39], 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 [29, 32]. Time-aware IM, which considers a time constraint during the diffusion process, is studied in [18, 30]. Competitive IM [10, 31] 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 [32], constraint to user groups [40], and under adaptive settings [23, 36]. These are orthogonal to our study.

7 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 200 GB memory.

Algorithms. 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 [3], [5] and [4], 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. We further implement two versions of HIST, one with vanilla RR set generation algorithm and one with SUBSIM framework for RR set generation. 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. We omit the result if the algorithm consumes more than 200GB memory.

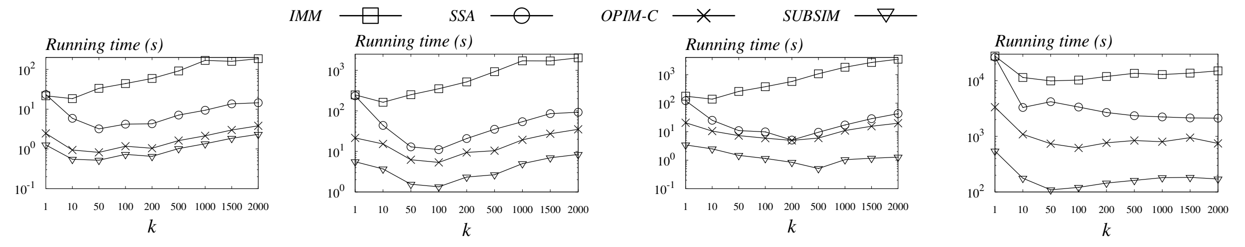
Table 2: Summary of datasets ($\mathbf{M = 10^6, B = 10^9}$)

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

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

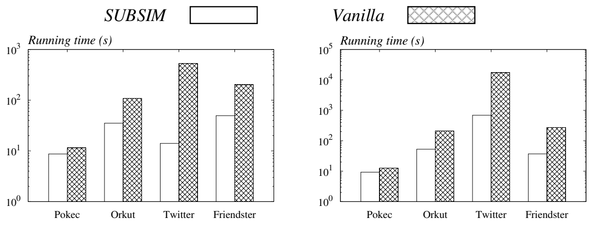
Parameter Settings. Recap that all the algorithms include an error parameter ϵ and a failure probability parameter δ. Following previous work [37], we set $\epsilon = 0.1$ and $\delta = 1/n$ for all solutions in the experiments. To examine the effectiveness of our SUBSIM, we compare our SUBSIM against 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 [38], 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 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.



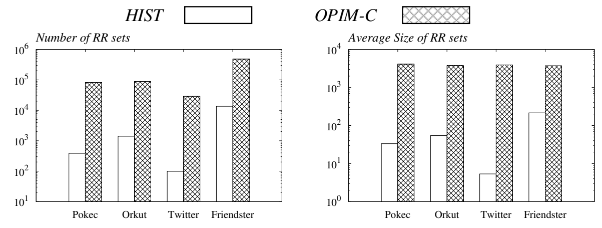
(a) Pokec (b) Orkut (c) Twitter (d) Friendster

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



(a) Exponential distribution (b) Weibull distribution

Figure 2: Skewed distribution: RR set generation cost.



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

Figure 3: Statistics of RR sets.

7.1 Effectiveness of SUBSIM

In the first set of experiments, we examine the effectiveness of SUBSIM against IMM, SSA, and OPIM-C 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.

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. We generate $2^{10} \times 1000$ random RR sets on each dataset using the vanilla algorithm and our SUBSIM, and report their running time. As shown in Figure 2, SUBSIM consistently keeps its advantage on all four tested datasets and achieves up to 38x (resp. 25x) speedup over vanilla under exponential (resp. Weibull) distribution.

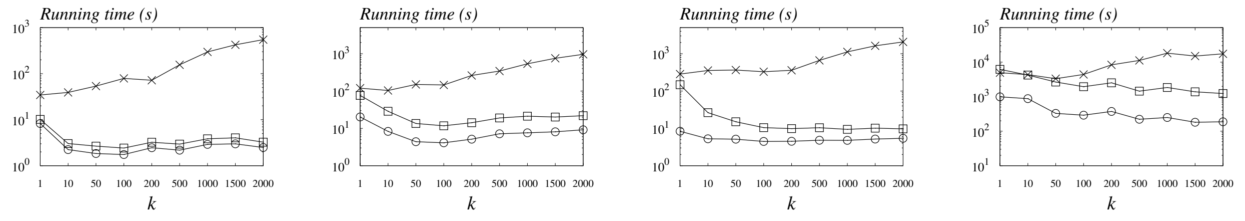
7.2 Effectiveness of HIST

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 4 shows the average running time of HIST (with vanilla RR set generation algorithm), HIST+SUBSIM (with SUBSIM for RR set generation), and OPIM-C. 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. Figure 5 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 6 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, SUBSIM+HIST is always two orders of magnitude faster than OPIM-C when $\theta = \theta_{32K}$.

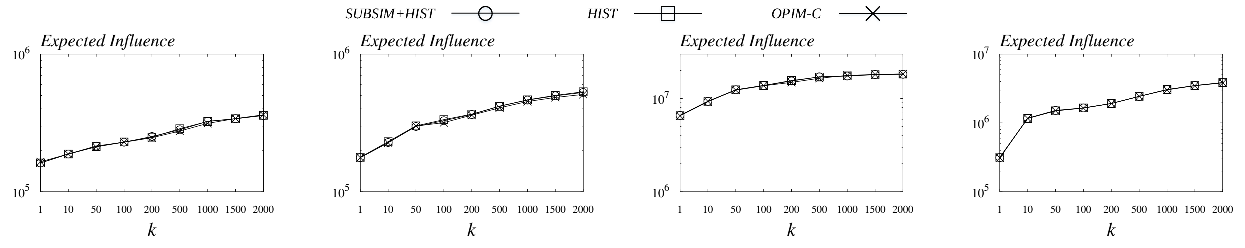
In our third set of experiments, we vary the average size of random RR sets under Uniform IC setting. We fix $k = 200$ and vary p with $\{p_{50}, p_{400}, p_{1K}, p_{4K}, p_{8K}, p_{32K}\}$. Figure 7 shows the running time of all three algorithms. We can see that even when the average size of RR sets 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. 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 the last set of experiments, we report some statistics of RR sets with our HIST under WC variant setting with



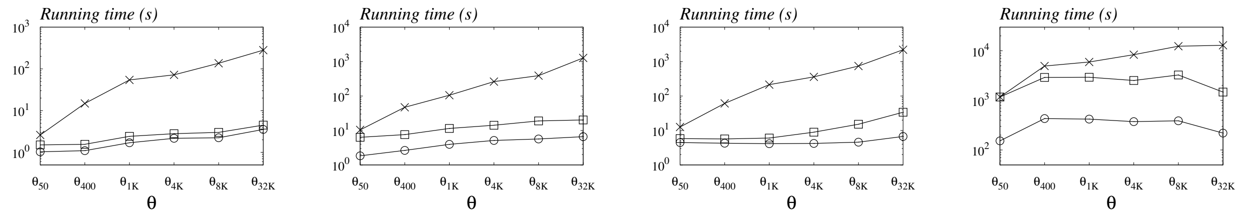
(a) Pokec (b) Orkut (c) Twitter (d) Friendster

Figure 4: Varying k: Running time under WC variant setting.



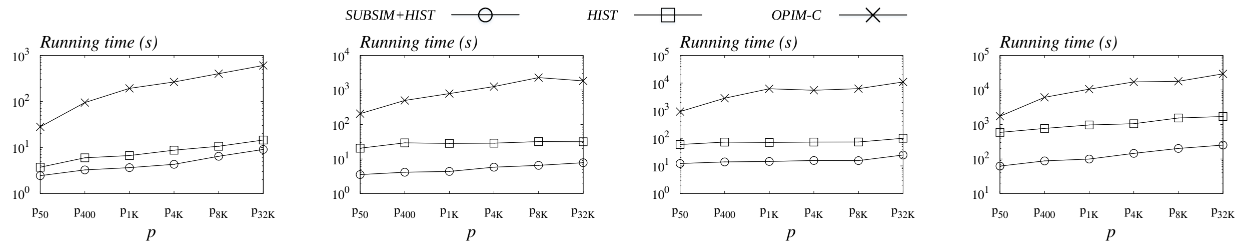
(a) Pokec (b) Orkut (c) Twitter (d) Friendster

Figure 5: Varying k: Expected influence under WC variant setting.



(a) Pokec (b) Orkut (c) Twitter (d) Friendster

Figure 6: Varying θ: Running time under WC variant setting.



(a) Pokec (b) Orkut (c) Twitter (d) Friendster

Figure 7: Varying p: Running time under Uniform IC setting.

$k = 2000$ and $\theta = \theta_{4k}$. Figure 3(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 3(b) reports the average size of random RR sets generated by our HIST against OPIM-C. Observe that the average size of 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 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 CONCLUSION

This paper presents SUBSIM, an efficient framework for RR set generation. We further present HIST to further tackle the challenging scalability issues in high influence networks.

9 ACKNOWLEDGMENTS

Sibo Wang is supported by Hong Kong RGC ECS Grant No. 24203419, CUHK Direct Grant No. 4055114, and a CUHK University Startup Grant. Zhewei Wei is supported by NSFC Grant No. 61832017, No. 61972401 and No. 61932001.

REFERENCES

[1] 2013. KONECT Datasets. http://konect.uni-koblenz.de/.

[2] 2014. SNAP Datasets. http://snap.stanford.edu/data.

[3] 2015. IMM code. https://sourceforge.net/projects/im-imm/.

[4] 2017. OPIM-C code. https://github.com/tangj90/OPIM.

[5] 2017. SSA code. https://github.com/hungnt55/Stop-and-Stare.

[6] 2020. SUMSIM technical report. https://sites.google.com/site/sigmod2020subsimtr/.

[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] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *SODA*. 946–957.

[9] Karl Bringmann and Konstantinos Panagiotou. 2017. Efficient Sampling Methods for Discrete Distributions. *Algorithmica* 79, 2 (2017), 484–508.

[10] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. In *WWW*. 665–674.

[11] 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.

[12] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*. 1029–1038.

[13] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *SIGKDD*. 199–208.

[14] Suqi Cheng, Huawei Shen, Junming Huang, Wei Chen, and Xueqi Cheng. 2014. IMRank: influence maximization via finding self-consistent ranking. In *SIGIR*. 475–484.

[15] 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.

[16] 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.

[17] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic Influence Maximization: Combining Scalability and Efficiency with Opinion-Aware Models. In *SIGMOD*. 743–758.

[18] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. 2011. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML*. 561–568.

[19] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2010. Learning influence probabilities in social networks. In *WSDM*. 241–250.

[20] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2011. A Data-Based Approach to Social Influence Maximization. *PVLDB* 5, 1 (2011), 73–84.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

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

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

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

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

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

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

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

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

[42] 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.