Online Processing Algorithms for Influence Maximization

Jing Tang
School of Computer Science and Engineering
Nanyang Technological University, Singapore
tang0311@ntu.edu.sg

Xueyan Tang
School of Computer Science and Engineering
Nanyang Technological University, Singapore
asxytang@ntu.edu.sg

Xiaokui Xiao*
School of Computing
National University of Singapore
xkxiao@nus.edu.sg

Junsong Yuan
Computer Science and Engineering Department
State University of New York at Buffalo
jsyuan@buffalo.edu

ABSTRACT

Influence maximization is a classic and extensively studied problem with important applications in viral marketing. Existing algorithms for influence maximization, however, mostly focus on *offline processing*, in the sense that they do not provide any output to the user until the final answer is derived, and that the user is not allowed to terminate the algorithm early to trade the quality of solution for efficiency. Such lack of interactiveness and flexibility leads to poor user experience, especially when the algorithm incurs long running time.

To address the above problem, this paper studies algorithms for *online processing* of influence maximization (OPIM), where the user can pause the algorithm at any time and ask for a solution (to the influence maximization problem) and its approximation guarantee, and can resume the algorithm to let it improve the quality of solution by giving it more time to run. (This interactive paradigm is similar in spirit to online query processing in database systems.) We show that the only existing algorithm for OPIM is vastly ineffective in practice, and that adopting existing influence maximization methods for OPIM yields unsatisfactory results. Motivated by this, we propose a new algorithm for OPIM with both superior empirical effectiveness and strong theoretical guarantees, and we show that it can also be extended to handle conventional influence maximization. Extensive experiments on real data demonstrate that our solutions outperform the state of the art for both OPIM and conventional influence maximization.

CCS CONCEPTS

• **Information systems** → *Data mining*; *Social advertising*; *Social networks*; • **Theory of computation** → *Probabilistic computation*; *Submodular optimization and polymatroids*;

KEYWORDS

Influence Maximization; Online Processing Algorithm; Sampling

ACM Reference Format:
Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online Processing Algorithms for Influence Maximization. In *Proceedings of 2018 SIGMOD International Conference on Management of Data (SIGMOD'18).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3183713.3183749

1 INTRODUCTION

Leveraging the word-of-mouth effects, viral marketing in online social networks (OSNs) has become an effective promotion channels for various products or campaigns [10]. Nowadays, the market of OSN advertisement is growing consistently due to the large number of users in OSNs. For example, Fortune [31] estimates that around 600 million US dollars of advertisement spending went to social media for the 2016 US president election.

To make a viral marketing campaign successful, it is crucial that a set of influential OSN users can serve as *seeds* to spread the marketing information to create a large information cascade. In addition, the number of seed users should be upper bounded by a constant k, due to avoid excessive marketing costs. Kempe et al. [20] are the first to formulate this problem as a constraint optimization problem referred to as *influence maximization*. The problem takes as input an OSN and a constant k, and asks for a set of k seed users with the largest expected size of influence cascade. Kempe et al. [20] show that influence maximization is NP-hard in general, and propose a simple and elegant approximation algorithm for a few fundamental classes of influence diffusion models, include the *independent cascade (IC)* [20] and *linear threshold (LT)* [20] models. Since then, a plethora of techniques have been proposed to improve the efficiency or effectiveness of influence maximization [1, 2, 4–9, 11–15, 18, 19, 22, 24, 28, 30, 32, 34, 35, 38, 39, 44, 45].

Almost all existing algorithms on influence maximization, however, focus on *offline* processing. Specifically, the state-of-the-art solutions [2, 18, 28, 38, 39] require the user to provide as input a desired approximation guarantee (on the quality of the seed set returned), and they may take a relatively long time to run when the approximation requirement is stringent. During this process, the user does not receive any indicator on the progress of the algorithm, and is forced to wait until the algorithm generates the final output. In addition, if the user chooses to terminate the algorithm early (e.g., due to long waiting time), she would not obtain any partial

*Work partially done when the author was with NTU, Singapore.

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3183749

result. Such lack of interactiveness and flexibility often leads to poor user experience.

To address the above problem, this paper studies algorithms for *online* processing of influence maximization (OPIM). In particular, an OPIM algorithm does not require users to specify the desired approximation guarantee in advance. Instead, at any timestamp t, the user is allowed to inquire about the algorithm's process, and the algorithm would provide the user an indicator of the approximation assurance that it can achieve so far. If the user is satisfied with the current approximation guarantee α, then she can terminate the algorithm and obtain a seed set with α-approximation; otherwise, she can let the algorithm continue running to improve the quality of the solution. This interactive processing paradigm is similar in spirit to *online query processing* [16, 17] in database systems, whereby the database user (i) submits a (potentially long running) query and receives approximate answers whose accuracies increase as time evolves, and (ii) has full control on when the query can be stopped.

To our knowledge, there is only one existing algorithm that is specifically designed for OPIM [2]. The algorithm utilizes a simple and elegant sampling technique referred to as *reverse influence sampling*, and it provides non-trivial approximation guarantees. However, the algorithm is rather pessimistic in its derivation of approximate solutions from samples. As a consequence, it offers poor empirical results and is of theoretical interests only, as we demonstrate in Section 8. On the other hand, although it is possible to adopt the state-of-the-art influence maximization algorithms [18, 28, 38, 39] to address OPIM, the adoption yields unsatisfactory efficiency, as we illustrate in Sections 3.3 and 8. This is due to the fundamental differences between OPIM and conventional influence maximization: the former optimizes the approximation guarantee of the seed set returned under a fixed budget of processing time, while the latter aims to minimize the computation overhead in achieving a predefined approximation ratio, i.e., OPIM *swaps* the constraint and optimization goal in influence maximization.

Motivated by the deficiency of the existing solutions, we present a new algorithm for the OPIM problem that provides superior empirical results as well as non-trivial worst-case guarantees. In particular, our algorithm can be paused by the user at any timestamp and then produce a seed set S and report an approximation guarantee α, such that S is an α-approximate solution of the influence maximization problem with at least $1-\delta$ probability, where $\delta \in (0,1)$ is a user-specified parameter. Compared with the existing OPIM method [2], our solution reports a much tighter approximation guarantee α given the same amount of time, since it utilizes an *instance-specific* algorithm to assess the quality assurance of S without resorting to the overly-pessimistic approach used in [2]. Furthermore, we show that our OPIM method can be extended to address conventional influence maximization (with offline processing), and it even outperforms the state-of-the-art solutions because of the advanced algorithm that we used to derive α.

More specifically, we make the following contributions.

(1) We analyze the deficiencies of the existing algorithms, and propose an OPIM algorithm that returns α-approximate solutions with at least $1-\delta$ probability at any given timestamp. (Sections 3 and 4)

(2) We devise a novel approach to improve the approximation guarantee α reported by our algorithm, and show that it is theoretically robust and does not degrade the efficiency of our OPIM algorithm. (Section 5)

(3) We present an extension of our OPIM algorithm to address conventional influence maximization, and prove that its efficiency and accuracy guarantees match those of the best existing influence maximization algorithms. (Section 6)

(4) We conduct extensive experiments to evaluate the performance of our algorithms against the state of the art, using datasets with up to 1.5 billion edges. Our results demonstrate that, given the same amount of computation time, our OPIM algorithm returns a solution whose reported approximation guarantee is significantly better than those reported by both the OPIM method in [2] and the OPIM-adoptions of existing influence maximization algorithms. Furthermore, for the conventional influence maximization problem, our solution can be orders of magnitude faster than the state of the art when offering the same theoretical guarantees. (Section 8)

2 PRELIMINARIES

2.1 Influence Maximization

We abstract an OSN as a directed graph $G = (V, E)$ with a set V of nodes (representing users) and a set E of edges (representing connections among users). For each directed edge $\langle u, v\rangle \in E$, we say that v is an *out-neighbor* of u, and u is an *in-neighbor* of v.

We focus on two basic and widely adopted diffusion models, i.e., the *independent cascade (IC)* and *linear threshold (LT)* models [20]. In both models, each edge $\langle u, v\rangle$ in G is associated with a *propagation probability* $p(u, v)$, representing the probability that u can influence v. Given a set of *seed* nodes S, both models consider the influence diffusion from S as an iterative process as follows. Initially, the seed nodes S are *activated* and the other nodes are *inactive*. When a node u first becomes activated at timestamp i, it has a *single* chance to activate its inactive out-neighbors at timestamp $i+1$; after that u cannot activate any other nodes. This process repeats until no more node can be activated. The difference between the IC and LT models lies in the details of node activation:

- *IC model.* Suppose that at timestamp i, a node u first becomes activated, and one of its out-neighbors v is inactive. Then, at timestamp $i+1$, u activates v with a probability $p(u, v)$.
- *LT model.* The LT model assumes that for each node v, (i) the propagation probabilities of v's incoming edges have a sum no more than 1, and (ii) there exits a threshold λ_v 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 denotes the set of in-neighbors of v that are activated at timestamp i.

Let $\sigma(S)$ denote the expected number of nodes activated by a seed set S via an influence diffusion process. We refer to $\sigma(S)$ as the *expected spread* of the seed set S. Given a graph G, an influence diffusion model, and a positive integer k, the *influence maximization* problem [20] aims to identify a size-k seed set S that maximizes $\sigma(S)$.

Kempe et al. [20] show that the influence maximization problem is NP-hard in general, and propose a greedy algorithm that returns a seed set whose expected spread is at least $1 - 1/e - \varepsilon$ fraction of the optimal expected spread with high probability, where ε is a user-specified parameter. Subsequently, a number of algorithms have been proposed to achieve the same approximation guarantee with improved efficiency [2, 18, 28, 38, 39].

2.2 Problem Definition

As mentioned in Section 1, we consider an alternative formulation of the influence maximization problem proposed by Borgs et al. [2], where the user (i) does not specify the desired approximation guarantee in advance, but (ii) keeps the influence maximization algorithm running and monitors its progress, and stops the algorithm at a certain timestamp and asks for a seed set S and its approximation assurance. We refer to this problem as *online processing of influence maximization (OPIM)*, and we formalize it as follows. Given an OSN G, an influence diffusion model, an integer k, a failure probability δ, and a streaming sequence of timestamps $t_1, t_2, t_3, \ldots$, we ask for a seed set S_i at timestamp t_i and an approximation guarantee α_i, such that S_i achieves α_i-approximation with at least $1 - \delta$ probability. Our objective is to maximize $\alpha_1, \alpha_2, \alpha_3, \ldots$.

The main difference between the conventional influence maximization problem and OPIM is that the former aims to minimize the time required to achieve a predefined approximation guarantee, whereas the latter aims to continuously optimize approximation guarantees as time evolves. This renders the algorithms for conventional influence maximization unsuitable for OPIM, as we discuss in Section 3.

3 EXISTING SOLUTIONS REVISITED

In this section, we (i) review the only existing solution [2] for OPIM, and (ii) discuss how the state-of-the-art algorithms [18, 28, 38, 39] for conventional influence maximization can be adopted for OPIM. Section 3.1 revisits *reverse influence sampling* [2], which is a crucial technique used in [2, 18, 28, 38, 39]. After that, Sections 3.2 and 3.3 present detailed discussions about existing algorithms.

3.1 Reverse Influence Sampling

Reverse influence sampling (RIS) [2] is a technique for estimating the expected spreads of seed sets, and it utilizes subgraph samples from G that are referred to as *random reverse reachable (RR) sets*. Generally speaking, a random RR set R is constructed in two steps:

(1) Select a node v from V uniformly at random.

(2) Produce a sample set R of the nodes in V, such that for each node $u \in V$, the probability that it appears in R equals the probability that a singleton seed set $\{u\}$ can activate v in an influence diffusion process.

Previous work [2, 38, 39] has shown that random RR sets can be efficiently constructed under a number of information diffusion models, including IC and LT. (Interested readers are referred to Appendix A for details.) In addition, Borgs et al. establish the following result.

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

Input: A set $\mathcal{R}$ of random RR sets and the seed set size k
Output: A size-k seed set S^*

```
initialize S* ← ∅;
for i ← 1 to k do
    u ← arg max_{v ∈ V \ S*} Λ(v | S*);
    S* ← S* ∪ {u};
return S*;
```

Lemma 3.1 ([2]). *Let $S \subseteq V$ be a fixed seed set and R be a random RR set. Then,*

$$\sigma(S) = |V| \cdot \Pr[S \cap R \neq \emptyset]. \quad (1)$$

By Lemma 3.1, we can estimate the expected spread of a seed set S using random RR sets. In particular, given a set $\mathcal{R}$ of random RR sets, we say that S *covers* an RR set $R \in \mathcal{R}$ if $S \cap R \neq \emptyset$, and we define the *coverage* of S in $\mathcal{R}$, denoted as $\Lambda(S)$, as the number of RR sets in $\mathcal{R}$ that are covered by S. Then, according to Lemma 3.1, $\frac{|V|}{|\mathcal{R}|}\Lambda(S)$ is an unbiased estimation of S's expected spread $\sigma(S)$.

Based on the above result, Borgs et al. propose a general framework for influence maximization that consists of two steps. First, we generate a set $\mathcal{R}$ of random RR sets; Then, we apply a standard greedy method, as shown in Algorithm 1, to identify a size-k seed set S^* with large coverage in $\mathcal{R}$. Specifically, the greedy algorithm first initializes $S^* = \emptyset$, and then iteratively inserts k nodes into S^*. In the i-th iteration ($1 \leq i \leq k$), it examines the nodes currently not in S^*, and inserts into u that maximizes $\Lambda(v \mid S^*)$, which is defined as:

$$\Lambda(v \mid S^*) = \Lambda(S^* \cup \{v\}) - \Lambda(S^*).$$

That is, $\Lambda(v \mid S^*)$ equals the number of RR sets in S^* that are covered by $\{v\}$ but not S^*. We refer to $\Lambda(v \mid S^*)$ as the *marginal coverage* of v given S^*.

By the standard result on the greedy method, Algorithm 1 returns a seed set S^* whose coverage in $\mathcal{R}$ is at least $1 - 1/e$ fraction of the coverage of any other size-k seed set. Accordingly, Borgs et al. show that S^* is a $(1 - 1/e - \varepsilon)$-approximate solution for influence maximization with at least $1 - \delta$ probability, if the total number of edges in G examined during the construction of $\mathcal{R}$ is $O(k(n + m)\varepsilon^{-3}(\ln(1/\delta))^2)$, where $n = |V|$ and $m = |E|$ are the total number of nodes and edges in G, respectively. Subsequent work [18, 28, 38, 39] shows that the same approximation guarantee can be achieved by imposing requirements on $\mathcal{R}$ to facilitate more efficient algorithms.

3.2 Borgs et al.'s Algorithm for OPIM

Borgs et al. [2] propose the first algorithm for OPIM with non-trivial approximation guarantee. The algorithm keeps generating random RR sets, and monitors the total number γ of edges examined during the RR set construction. Whenever γ equals a power of 2, the algorithm pauses and derives a seed set S^* by applying the greedy method (i.e., Algorithm 1) on the set $\mathcal{R}$ of RR sets that has been produced. In addition, it calculates the approximation guarantee of S^* as $\min\{1/4, \beta\}$, where

$$\beta = \frac{\gamma}{1492992 \cdot (n + m) \ln n}.$$

This approximation guarantee is correct with a probability only 3/5, but this probability can be improved to $1-\delta$ for any $0 < \delta < 2/5$, at the cost of increasing the computation time of the algorithm by a factor of $O(\log(1/\delta))$. Whenever the user asks for a seed set, Borgs et al.'s algorithm retrieves the last seed set generated when γ was a power of 2, and returns it along with the approximation guarantee calculated then.

Borgs et al.'s OPIM algorithm is simple and elegant from a theoretical perspective, but is insufficient for practical applications, since the approximation guarantee that it reports is extremely loose. For example, suppose that we are to obtain a 0.1-approximate solution on a graph with $n = 10^5$ nodes and $m = 10^6$ edges. Then, Borgs et al.'s algorithm would require more than 2×10^{12} edges to be examined, which incurs prohibitive overheads. Furthermore, the approximation ratio reported by the algorithm is capped at 1/4, regardless of the value of γ. The main reason for these deficiencies is that the algorithm evaluates the approximation guarantee of S^* solely based on γ, m, and n, without utilizing any instance-specific information in S^*; as a consequence, it is often overly pessimistic when assessing S^*'s quality assurance, which severely degrades its practical performance.

3.3 Adopting Influence Maximization Algorithms for OPIM

Instead of using Borgs et al.'s OPIM algorithm, we may adopt the state-of-the-art RIS-based influence maximization algorithms [18, 28, 38, 39] for OPIM. Specifically, suppose that we have an influence maximization algorithm $\mathcal{A}$ that returns a $(1-1/e-\varepsilon)$-approximation with at least $1-\delta$ probability. Then, we repeatedly invoke $\mathcal{A}$ to generate seed sets, such that the i-th ($i \geq 1$) invocation is with ε set to $(1-1/e)/2^{i-1}$. (Note that we start with $\varepsilon = 1-1/e$, since any larger ε renders the $(1-1/e-\varepsilon)$-approximation guarantee meaningless.) If the user asks for a seed set while we are conducting the j-th execution of $\mathcal{A}$, then we retrieve the seed set produced by the $(j-1)$-th execution, return it to the user, and report $(1-1/e)(1-1/2^{j-2})$ as its approximation ratio.

The above solution provides better empirical results than Borgs et al.'s (largely theoretical) approach, as we demonstrate in Section 8. However, it still leaves much room for improvement, for two reasons. First, when a user asks for a seed set at timestamp t, it cannot fully utilize the $\mathcal{R}$ of all RR sets generated before t, but has to resort to a seed set generated at an earlier timestamp (when $\mathcal{R}$ might be much smaller). This may render the quality of solution less than satisfactory. Second, the algorithm requires multiple executions of $\mathcal{A}$, each of which generates a number of RR sets; yet, the seed set returned to the user is only based on the RR sets constructed during the last execution, i.e., the RR sets constructed during all executions before that are wasted.

To address the above issues, a natural idea is to ask $\mathcal{A}$ to use the set $\mathcal{R}$ of all RR sets generated before timestamp t to compute a seed set S^*, and then derive S^*'s approximation guarantee from $\mathcal{R}$ and report it to the user. This, however, requires ε to be derived based on $\mathcal{R}$, which is infeasible with the existing RIS-based algorithms [18, 28, 38, 39]. The reason is that the theoretical analyses in [18, 28, 38, 39] all assume that ε is given in advance and is independent of $\mathcal{R}$; when this assumption does not hold, the theoretical analyses are no longer valid, as discussed in Section 4.3 of previous work [18]. This motivates us to design an OPIM algorithm based on new analysis that avoids the assumption on fixed ε, which we elaborate in Section 4.

4 OUR SOLUTION

4.1 Overview

Our solution for OPIM is similar in spirit to Borgs et al.'s algorithm [2], in that it also utilizes reverse influence sampling in a streaming fashion. In a nutshell, our solution continuously generates RR sets on the input graph G; whenever it is asked to produce a seed set S, it uses the RR sets generated to derive S as well as its approximation guarantee.

Specifically, given the set $\mathcal{R}$ of RR sets that have been generated, our solution derives a seed set S^* and its quality assurance in three steps. First, it evenly divides $\mathcal{R}$ into two subsets $\mathcal{R}_1$ and $\mathcal{R}_2$. Then, it identifies S^* based on the RR sets in $\mathcal{R}_1$. Finally, it uses $\mathcal{R}_2$ to compute the approximation guarantee of S^*. The derivation of S^* from $\mathcal{R}_1$ is a direct application of Borgs et al.'s greedy algorithm described in Section 3.2. That is, we start with $S^* = \emptyset$, and then iteratively add into S^* the node with the largest marginal coverage in $\mathcal{R}_1$, until the size of S^* reaches k.

The computation of S^*'s approximation guarantee, however, is drastically different from Borgs et al.'s approach (as described in Section 3.2). In particular, we do not derive S^*'s quality assurance based only on the number of RR sets; instead, we use $\mathcal{R}_1$ and $\mathcal{R}_2$ to compute a lower bound $\sigma^l(S^*)$ of S^*'s expected spread and an upper bound $\sigma^u(S^o)$ of the optimal seed set S^o's expected spread, and then we return $\sigma^l(S^*)/\sigma^u(S^o)$ as the approximation guarantee of S^*. In Section 4.2, we will explain our derivation of $\sigma^l(S^*)$ and $\sigma^u(S^o)$.

4.2 Derivations of $\sigma^l(S^*)$ and $\sigma^u(S^o)$

To compute $\sigma^l(S^*)$ and $\sigma^u(S^o)$, we utilize two martingale-based concentration bounds from [38] that are tailored for influence estimation based on RR sets:

Lemma 4.1 ([38]). *Given a fixed number of θ random RR sets $\mathcal{R}$ and a seed set S, let $\Lambda(S)$ be the coverage of S in $\mathcal{R}$. For any $\lambda > 0$,*

$$\Pr\left[\Lambda(S) - \sigma(S)\cdot\frac{\theta}{n} \geq \lambda\right] \leq \exp\left(-\frac{\lambda^2}{2\sigma(S)\cdot\frac{\theta}{n} + \frac{2}{3}\lambda}\right), \quad (2)$$

$$\Pr\left[\Lambda(S) - \sigma(S)\cdot\frac{\theta}{n} \leq -\lambda\right] \leq \exp\left(-\frac{\lambda^2}{2\sigma(S)\cdot\frac{\theta}{n}}\right). \quad (3)$$

Intuitively, Lemma 4.1 provides us a way to quantify how much $\Lambda(S)$ may deviate from $\sigma(S)\cdot\frac{\theta}{n}$ (i.e., the expected coverage of S in $\mathcal{R}$) with a given probability. In the following, we will derive $\sigma^l(S^*)$ and $\sigma^u(S^o)$ based on Lemma 4.1.

Derivation of $\sigma^l(S^*)$. Let S^* and $\mathcal{R}_2$ be as defined in Section 4.1. Let $\Lambda_2(S^*)$ be the coverage of S in $\mathcal{R}_2$, and $\theta_2 = |\mathcal{R}_2|$. We have the following result.

Lemma 4.2. *For any $\delta \in (0,1)$, we have*

$$\Pr\left[\sigma(S^*) \geq \left(\left(\sqrt{\Lambda_2(S^*) + \frac{2a}{9}} - \sqrt{\frac{a}{2}}\right)^2 - \frac{a}{18}\right) \cdot \frac{n}{\theta_2}\right] \geq 1-\delta, \quad (4)$$

where $a = \ln(1/\delta)$.

Proof.

$$\begin{aligned}
&\Pr\left[\sigma(S^*) < \left(\left(\sqrt{\Lambda_2(S^*) + \frac{2a}{9}} - \sqrt{\frac{a}{2}}\right)^2 - \frac{a}{18}\right) \cdot \frac{n}{\theta_2}\right] \\
&= \Pr\left[\sqrt{\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a}{18}} < \sqrt{\Lambda_2(S^*) + \frac{2a}{9}} - \sqrt{\frac{a}{2}}\right] \\
&\quad + \Pr\left[\sqrt{\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a}{18}} < \sqrt{\frac{a}{2}} - \sqrt{\Lambda_2(S^*) + \frac{2a}{9}}\right] \\
&= \Pr\left[\sqrt{\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a}{18}} < \sqrt{\Lambda_2(S^*) + \frac{2a}{9}} - \sqrt{\frac{a}{2}}\right] \\
&= \Pr\left[\sqrt{2a\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a^2}{9}} + \frac{a}{3} < \Lambda_2(S^*) - \frac{\sigma(S^*)\cdot\theta_2}{n}\right] \\
&\leq \exp\left(-\frac{\left(\sqrt{2a\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a^2}{9}} + \frac{a}{3}\right)^2}{2\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{2}{3}\cdot\left(\sqrt{2a\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a^2}{9}} + \frac{a}{3}\right)}\right) \\
&= \exp(-a) \\
&= \delta,
\end{aligned}$$

where the second equality is due to

$$\sqrt{\frac{\sigma(S^*)\cdot\theta_2}{n} + \frac{a}{18}} \geq \sqrt{\frac{a}{18}} = \sqrt{\frac{a}{2}} - \sqrt{\frac{2a}{9}} \geq \sqrt{\frac{a}{2}} - \sqrt{\Lambda_2(S^*) + \frac{2a}{9}},$$

and the inequality is due to (2). Hence, the lemma is proved. □

Based on Lemma 4.2, we set

$$\sigma^l(S^*) = \left(\left(\sqrt{\Lambda_2(S^*) + \frac{2\ln(1/\delta_2)}{9}} - \sqrt{\frac{\ln(1/\delta_2)}{2}}\right)^2 - \frac{\ln(1/\delta_2)}{18}\right) \cdot \frac{n}{\theta_2}, \quad (5)$$

where $\delta_2 \in (0, \delta)$ is a parameter that we will discuss shortly.

Derivation of $\sigma^u(S^o)$. Computing an upper bound of $\sigma(S^o)$ may seem challenging at the first sight, since S^o (i.e, the optimal seed set) is unknown and NP-hard to identify. However, even without knowing S^o, we can still establish an upper bound of $\sigma(S^o)$ based on the coverage of S^* in $\mathcal{R}_1$, denoted as $\Lambda_1(S)$. The rationale is as follows. First, by the property of the greedy algorithm, we have

$$\Lambda_1(S^*) \geq \left(1 - \frac{1}{e}\right) \cdot \Lambda_1(S^o). \quad (6)$$

Meanwhile, by Lemma 4.1, $\Lambda_1(S^o) \geq x \cdot \sigma(S^o) + y$ holds with high probability, for certain values x and y depending on $\sigma(S^o)$. This leads to $\Lambda_1(S^*) \geq \left(1 - \frac{1}{e}\right) \cdot (x \cdot \sigma(S^o) + y)$, which provides us a means to bound $\sigma(S^o)$ from above. More formally, we have the following lemma.

Lemma 4.3. *Let $\theta_1 = |\mathcal{R}_1|$. For any $\delta \in (0,1)$,*

$$\Pr\left[\sigma(S^o) \leq \left(\sqrt{\frac{\Lambda_1(S^*)}{1-1/e} + \frac{a}{2}} + \sqrt{\frac{a}{2}}\right)^2 \cdot \frac{n}{\theta_1}\right] \geq 1-\delta, \quad (7)$$

where $a = \ln(1/\delta)$.

Proof. By (6) and Lemma 4.1,

$$\begin{aligned}
&\Pr\left[\sigma(S^o) > \left(\sqrt{\frac{\Lambda_1(S^*)}{1-1/e} + \frac{a}{2}} + \sqrt{\frac{a}{2}}\right)^2 \cdot \frac{n}{\theta_1}\right] \\
&\leq \Pr\left[\sigma(S^o) > \left(\sqrt{\Lambda_1(S^o) + \frac{a}{2}} + \sqrt{\frac{a}{2}}\right)^2 \cdot \frac{n}{\theta_1}\right] \\
&\leq \Pr\left[\left(\sqrt{\frac{\theta_1\cdot\sigma(S^o)}{n}} - \sqrt{\frac{a}{2}}\right)^2 > \Lambda_1(S^o) + \frac{a}{2}\right] \\
&= \Pr\left[-\sqrt{2a\cdot\frac{\theta_1\cdot\sigma(S^o)}{n}} > \Lambda_1(S^o) - \frac{\theta_1\cdot\sigma(S^o)}{n}\right] \\
&\leq \exp\left(-\frac{2a\cdot\frac{\theta_1\cdot\sigma(S^o)}{n}}{2\cdot\frac{\theta_1\cdot\sigma(S^o)}{n}}\right) \\
&= \delta,
\end{aligned}$$

Therefore, the lemma is proved. □

Based on Lemma 4.3, we set

$$\sigma^u(S^o) = \left(\sqrt{\frac{\Lambda_1(S^*)}{1-1/e} + \frac{\ln(1/\delta_1)}{2}} + \sqrt{\frac{\ln(1/\delta_1)}{2}}\right)^2 \cdot \frac{n}{\theta_1}, \quad (8)$$

where $\delta_1 \in (0, \delta)$ is a parameter to be discussed shortly.

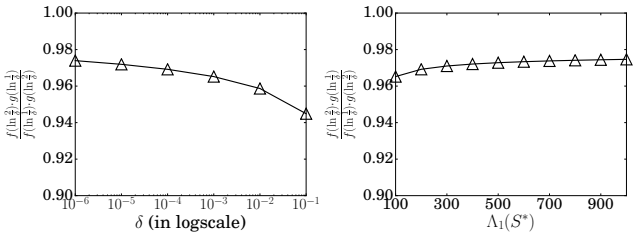
Putting It Together. In summary, we derive $\sigma^l(S^*)$ and $\sigma^u(S^o)$ based on (5) and (8), respectively, and we use $\frac{\sigma^l(S^*)}{\sigma^u(S^o)}$ as the approximation guarantee of S^*. By Lemmas 4.2 and 4.3, $\frac{\sigma^l(S^*)}{\sigma^u(S^o)}$ is correct with at least $1-\delta_1-\delta_2$ probability, where δ_1 and δ_2 are the parameters in (5) and (8), respectively. Therefore, it suffices to ensure that $\delta_1 + \delta_2 \leq \delta$. Accordingly, we set $\delta_1 = \delta_2 = \delta/2$. The following lemma shows how close this setting of δ_1 and δ_2 is to the optimal setting. The proof of the lemma is given in Appendix B.

Lemma 4.4. *Let α be the value of $\frac{\sigma^l(S^*)}{\sigma^u(S^o)}$ when $\delta_1 = \delta_2 = \delta/2$, and α' be the maximum value of $\frac{\sigma^l(S^*)}{\sigma^u(S^o)}$ for any $\delta_1, \delta_2 \in (0,\delta)$ satisfying $\delta_1 + \delta_2 \leq \delta$. Then,*

$$\frac{\alpha}{\alpha'} \geq \frac{f(\ln\frac{2}{\delta})\cdot g(\ln\frac{1}{\delta})}{f(\ln\frac{1}{\delta})\cdot g(\ln\frac{2}{\delta})},$$

where $f(x) = \left(\sqrt{\Lambda_2(S^*) + \frac{2x}{9}} - \sqrt{\frac{x}{2}}\right)^2 - \frac{x}{18}$ *and* $g(x) = \left(\sqrt{\frac{\Lambda_1(S^*)}{1-1/e} + \frac{x}{2}} + \sqrt{\frac{x}{2}}\right)^2$.

Figure 1 illustrates the value of $\frac{f(\ln\frac{2}{\delta})\cdot g(\ln\frac{1}{\delta})}{f(\ln\frac{1}{\delta})\cdot g(\ln\frac{2}{\delta})}$, under a representative setting of $\Lambda_2(S^*) = 100$ and varying δ and $\Lambda_1(S^*)$. Observe



(a) $\Lambda_1(S^*) = \Lambda_2(S^*) = 100$ **(b)** $\delta = 10^{-3}$, $\Lambda_2(S^*) = 100$

Figure 1: The value of $\frac{f(\ln\frac{2}{\delta})\cdot g(\ln\frac{1}{\delta})}{f(\ln\frac{1}{\delta})\cdot g(\ln\frac{2}{\delta})}$.

that $\frac{f(\ln\frac{2}{\delta})\cdot g(\ln\frac{1}{\delta})}{f(\ln\frac{1}{\delta})\cdot g(\ln\frac{2}{\delta})}$ is close to 1 in all cases, which demonstrates that $\delta_1 = \delta_2 = \delta/2$ is a near-optimal setting of the δ_1 and δ_2.

Time Complexity. The time required to compute S^* is $O(\sum_{R\in\mathcal{R}_1}|R|)$, since the greedy approach for maximum coverage runs in time linear to the total size of its input [40]. Moreover, computing $\sigma^l(S^*)$ and $\sigma^u(S^o)$ only takes time linear to $\sum_{R\in\mathcal{R}_1\cup\mathcal{R}_2}|R|$, namely, the total size of the RR sets in $\mathcal{R}_1$ and $\mathcal{R}_2$. This is because $\sigma^l(S^*)$ and $\sigma^u(S^o)$ only rely on $\Lambda_2(S^*)$ and $\Lambda_1(S^*)$, respectively, and $\Lambda_2(S^*)$ (resp. $\Lambda_1(S^*)$) can be derived via a linear scan of $\mathcal{R}_2$ (resp. $\mathcal{R}_1$). Therefore, the total time complexity of deriving S^* and its approximation guarantee is $O(\sum_{R\in\mathcal{R}_1\cup\mathcal{R}_2}|R|)$.

Discussions. Observe that our solution derives $\sigma^u(S^o)$ and $\sigma^l(S^*)$ from $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, instead of combining $\mathcal{R}_1$ and $\mathcal{R}_2$ together to compute both $\sigma^u(S^o)$ and $\sigma^l(S^*)$. The reason is that the derivation of $\sigma^l(S^*)$ relies on the concentration bound in Lemma 4.1, which requires S^* to be a fixed seed set independent of the RR sets on which $\sigma^l(S^*)$ is computed. To explain this in an intuitively manner, we may consider $\mathcal{R}_1$ as a set of *nominators* who nominate S^* as the solution to OPIM (since S^* is derived from $\mathcal{R}_1$), and $\mathcal{R}_2$ as a set of *judges* who determine whether S^* is good enough. If S^* is not independent of $\mathcal{R}_2$, then the judges' evaluation of S^* might be biased, in which case the quality assurance of S^* is questionable. Our solution does not suffer from such bias, since we ensure that $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$, i.e., the nominators for S^* are independent of the judges. (Interested readers are referred to Section 4.3 of previous work [18] for an illustration of the issues in using the same set of RR sets to both generate a seed set and evaluate its properties based on concentration bounds.)

In addition, when a user asks for seed sets at multiple timestamps, then our solution ensures that each seed set S^* returned provides an approximation guarantee of $\frac{\sigma^l(S^*)}{\sigma^u(S^o)}$ with at least $1-\delta$ probability. However, if one aims to guarantee that *all* seed sets returned achieve the claimed approximation guarantees *simultaneously* with at least $1-\delta$ probability, then our solution is insufficient, since each of our seed sets has up to δ probability to fail. To address this issue, we may vary the requirement on each seed set's failure probability. In particular, we can generate the i-th seed set such that its failure probability is at most $\delta/2^i$, so that the total failure probability is at most δ (by the union bound). For simplicity, however, we do not consider this variation in our paper.

5 IMPROVED GUARANTEES WITH TIGHTENED BOUNDS

The solution in Section 4 reports the approximation guarantee of each seed set S^* as $\frac{\sigma^l(S^*)}{\sigma^u(S^o)}$, which is relatively easy to compute but could be overly pessimistic. To explain, recall that the derivation of $\sigma^u(S^o)$ requires an upper bound of $\Lambda_1(S^o)$ (i.e., S^o's coverage in $\mathcal{R}_1$), for which we use $\Lambda_1(S^*)/(1-1/e)$ (see (6)). This upper bound, albeit tight in the worst case [27], is often loose for specific problem instances. Motivated by this, we propose another upper bound of $\Lambda_1(S^o)$ that is much tighter in practice, as explained in the following.

Let $\Lambda_1(v \mid S)$ be the marginal coverage of a node v in $\mathcal{R}_1$ with respect to a seed set S, and $maxMC(S,\ell)$ denote the set of ℓ nodes with the ℓ largest marginal coverage in $\mathcal{R}_1$ with respect to S. We observe that $\Lambda_1(S) + \sum_{v\in maxMC(S,k)}$ is an upper bound of $\Lambda_1(S^o)$, as shown in the following lemma.

Lemma 5.1. *For any seed set S with at most k nodes,*

$$\Lambda_1(S^o) \le \Lambda_1(S) + \sum_{v\in maxMC(S,k)} \Lambda_1(v \mid S). \tag{9}$$

Proof.

$$\begin{aligned}\Lambda_1(S^o) &\le \Lambda_1(S^o \cup S)\\ &\le \Lambda_1(S) + \sum_{v\in S^o\setminus S} \Lambda_1(v \mid S)\\ &\le \Lambda_1(S) + \sum_{v\in maxMC(S,|S^o\setminus S|)} \Lambda_1(v \mid S)\\ &\le \Lambda_1(S) + \sum_{v\in maxMC(S,k)} \Lambda_1(v \mid S),\end{aligned}$$

where the first two inequalities are due to the monotonicity and submodularity of $\Lambda_1(\cdot)$, respectively, and the third and forth inequalities are by the definition of $maxMC(S,k)$. □

Now consider the construction of S^* by running the greedy algorithm on $\mathcal{R}_1$. Let S_i^* ($i \in [1,k]$) be a set containing the i nodes that are selected in the first i iterations of the greedy algorithm, and let $S_0^* = \emptyset$. Then, by Lemma 5.1, we have a new upper bound of $\Lambda_1(S^o)$ as follows:

$$\Lambda_1^u(S^o) = \min_{0\le i\le k}\left(\Lambda_1(S_i^*) + \sum_{v\in maxMC(S_i^*,k)} \Lambda_1(v \mid S_i^*)\right). \tag{10}$$

The following lemma shows that this new upper bound of $\Lambda_1^u(S^o)$ is never worse than $\Lambda_1(S^*)/(1-1/e)$.

Lemma 5.2.

$$\Lambda_1^u(S^o) \le \frac{\Lambda_1(S^*)}{1-(1-1/k)^k} \le \frac{\Lambda_1(S^*)}{1-1/e}. \tag{11}$$

Proof. By the property of the greedy algorithm, for any $0 \le i \le k-1$, we have

$$\Lambda_1(S_{i+1}^*) - \Lambda_1(S_i^*) = \max_{v\in V} \Lambda_1(v \mid S_i^*).$$

Thus,

$$\begin{aligned}&\Lambda_1(S_i^*) + k \cdot \left(\Lambda_1(S_{i+1}^*) - \Lambda_1(S_i^*)\right)\\ &\geq \Lambda_1(S_i^*) + \sum_{v \in maxMC(S_i^*, k)} \Lambda_1(v \mid S_i^*)\\ &\geq \Lambda_1^u(S^o). \end{aligned} \tag{12}$$

Rearranging (12) yields

$$\Lambda_1^u(S^o) - \Lambda_1(S_{i+1}^*) \leq (1 - 1/k) \cdot \left(\Lambda_1^u(S^o) - \Lambda_1(S_i^*)\right).$$

Recursively, we have

$$\begin{aligned}\Lambda_1^u(S^o) - \Lambda_1(S_k^*) &\leq (1 - 1/k)^k \cdot \left(\Lambda_1^u(S^o) - \Lambda_1(S_0^*)\right)\\ &= (1 - 1/k)^k \cdot \Lambda_1^u(S^o).\end{aligned}$$

It follows that

$$\Lambda_1^u(S^o) \leq \frac{\Lambda_1(S_k^*)}{1 - (1 - 1/k)^k} \leq \frac{\Lambda_1(S_k^*)}{1 - 1/e}.$$

Hence, the lemma is proved. □

Based on Lemma 5.2, we propose a new upper bound of $\sigma(S^o)$ that is empirically much tighter than $\sigma^u(S^o)$:

$$\hat{\sigma}^u(S^o) = \left(\sqrt{\Lambda_1^u(S^o) + \frac{\ln(1/\delta_1)}{2}} + \sqrt{\frac{\ln(1/\delta_1)}{2}}\right)^2 \cdot \frac{n}{\theta_1}, \tag{13}$$

By a proof similar to that of Lemma 4.3, we can show that

$$\Pr\left[\sigma(S^o) \leq \hat{\sigma}^u(S^o)\right] \geq 1 - \delta_1. \tag{14}$$

Time Complexity. The computation of $\hat{\sigma}^u(S^o)$ is more sophisticated than that of $\sigma^u(S^o)$, since the former requires deriving $\Lambda_1(S_i^*)$ and $\sum_{v \in maxMC(S_i^*, k)} \Lambda_1(v \mid S_i^*)$ for any $0 \leq i \leq k$, while the latter only relies on $\Lambda_1(S^*)$. Nonetheless, we observe that (i) both $\Lambda_1(S_i^*)$ and $\Lambda_1(v \mid S_i^*)$ for any v are available from the i-th iteration of the greedy algorithm (invoked on $\mathcal{R}_1$), and (ii) for any i, we can identify the nodes in $maxMC(S_i^*, k)$ in $O(n)$ time by using a linear-time selection algorithm to identify the k-th largest marginal coverage and then scanning through nodes to find those with marginal coverage no less than the k-th largest. Therefore, the total time required to compute S^* and $\hat{\sigma}^u(S^o)$ from $\mathcal{R}_1$ is $O(kn + \sum_{R \in \mathcal{R}_1} |R|)$. Recall that computing $\sigma^l(S^*)$ takes $O(\sum_{R \in \mathcal{R}_2} |R|)$ time. Therefore, the total time complexity of deriving S^* and its improved approximation guarantee via $\hat{\sigma}^u(S^o)$ is $O(kn + \sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} |R|)$. This complexity is identical to that of the vanilla OPIM algorithm when $kn \leq \sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} |R|$, which is often the case in practice.

Comparison with Previous Work [24]. In [24], Leskovec et al. propose a method to evaluate the quality of the solution returned by the greedy algorithm for maximum coverage. If we apply Leskovec et al.'s method in our setting, we can obtain a different upper bound of $\Lambda_1(S^o)$ as follows:

$$\Lambda_1^\diamond(S^o) = \Lambda_1(S^*) + \sum_{v \in maxMC(S^*, k)} \Lambda_1(v \mid S^*).$$

Based on this, we have an alternative upper bound of $\sigma^\diamond(S^o)$:

$$\sigma^\diamond(S^o) = \left(\sqrt{\Lambda_1^\diamond(S^o) + \frac{\ln(1/\delta_1)}{2}} + \sqrt{\frac{\ln(1/\delta_1)}{2}}\right)^2 \cdot \frac{n}{\theta_1}. \tag{15}$$

Table 1: Time Complexities of Our OPIM Algorithms.

Algorithm	Time Complexity
Vanilla OPIM	$O(\sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} \lvert R \rvert)$
Improved OPIM via $\hat{\sigma}^u(S^o)$	$O(kn + \sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} \lvert R \rvert)$
Improved OPIM via $\sigma^\diamond(S^o)$	$O(n + \sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} \lvert R \rvert)$

Note that this upper bound is never better than $\hat{\sigma}^u(S^o)$, since $\Lambda_1^\diamond(S^o) \geq \Lambda_1^u(S^o)$. In addition, for certain problem instances, $\Lambda_1^\diamond(S^o)$ can be even larger than $\Lambda_1(S^*)/(1-1/e)$, in which case $\sigma^\diamond(S^o)$ is an even looser upper bound than $\sigma^u(S^o)$. In Section 8, we will show that $\sigma^\diamond(S^o)$ leads to considerably worse results than $\hat{\sigma}^u(S^o)$ does.

Note that it takes $O(n)$ time to compute $\Lambda_1^\diamond(S^o)$ after obtaining S^*. Thus, the time complexity of deriving S^* and its improved approximation guarantee via $\sigma^\diamond(S^o)$ is $O(n + \sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} |R|)$. We summarize the time complexities of three versions of our OPIM method in Table 1.

6 EXTENSION TO CONVENTIONAL INFLUENCE MAXIMIZATION

This section extends our OPIM algorithm (described in Sections 4 and 5) to address the conventional influence maximization problem, where (i) we are given a graph G, a diffusion model, a seed set size k, an error threshold ε, and a failure probability threshold δ, and (ii) we aim to return a size-k seed set whose expected spread is at least $1 - 1/e - \varepsilon$ times the optimal expected spread with at least $1 - \delta$ probability. The basic idea of our extension is as follows. We invoke our OPIM algorithm to continuously generate RR sets on G, and we ask the algorithm to return a seed set whenever the number of RR sets constructed reaches a power of 2. Whenever the seed set S^* returned has an approximation guarantee at least $1 - 1/e - \varepsilon$, we output S^* and terminate the algorithm.

In what follows, we clarify the details of our solution and prove its approximation guarantee and time complexity. For that purpose, we first introduce a result from [38].

Lemma 6.1 ([38]). *Let $\mathcal{R}$ be a set of random RR sets and S^* be a size-k seed set generated by applying the greedy algorithm on $\mathcal{R}$. For fixed ε and δ, if*

$$|\mathcal{R}| \geq \frac{2n\left((1-1/e)\sqrt{\ln\frac{2}{\delta}} + \sqrt{(1-1/e)(\ln\binom{n}{k} + \ln\frac{2}{\delta})}\right)^2}{\varepsilon^2 k},$$

then S^* is a $(1 - 1/e - \varepsilon)$-approximate solution with at least $1 - \delta$ probability.

Lemma 6.1 provides us an upper bound on the number of RR sets required to ensure $1 - 1/e - \varepsilon$ approximation for influence maximization. Based on Lemma 6.1, we define a value $\theta_{\max}$:

$$\theta_{\max} = \frac{2n\left((1-1/e)\sqrt{\ln\frac{6}{\delta}} + \sqrt{(1-1/e)(\ln\binom{n}{k} + \ln\frac{6}{\delta})}\right)^2}{\varepsilon^2 k}, \tag{16}$$

which is an upper bound on the number of RR sets needed to guarantee $1-1/e-\varepsilon$ approximation with at least $1-\delta/3$ probability. In addition, we define another value

$$\theta_0 = \theta_{\max} \cdot \varepsilon^2 k / n, \tag{17}$$

Algorithm 2: OPIM-C($G, k, \varepsilon, \delta$)

Input: Graph G, seed set size k, error threshold ε, and failure probability threshold δ
Output: A k-size seed set S^* that provides an approximation guarantee of at least $(1 - 1/e - \varepsilon)$ with probability at least $1 - \delta$

1 initialize $\theta_{\max}$ and θ_0 by (16) and (17), respectively;
2 generate two sets $\mathcal{R}_1$ and $\mathcal{R}_2$ of random RR sets, with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_0$;
3 $i_{\max} \leftarrow \lceil \log_2 \frac{\theta_{\max}}{\theta_0} \rceil$;
4 **for** $i \leftarrow 1$ **to** $i_{\max}$ **do**
5 apply the greedy algorithm on $\mathcal{R}_1$ to generate a size-k seed set S^*;
6 compute $\sigma^l(S^*)$ and $\hat{\sigma}^u(S^o)$ by (5) and (13), respectively, setting $\delta_1 = \delta_2 = \delta/(3i_{\max})$;
7 $\alpha \leftarrow \sigma^l(S^*)/\hat{\sigma}^u(S^o)$;
8 **if** $\alpha \geq 1 - 1/e - \varepsilon$ **or** $i = i_{\max}$ **then return** S^*;
9 double the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ with new random RR sets;

the usage of which will be clarified shortly.

Algorithm 2 shows the pseudo-code of our OPIM-based method for conventional influence maximization (*OPIM-C*). It first generates two sets of random RR sets $\mathcal{R}_1$ and $\mathcal{R}_2$, each of size θ_0 (Line 1). The subsequent part of the algorithm consists of at most $i_{\max} = \lceil \log_2 \frac{\theta_{\max}}{\theta_0} \rceil$ iterations (Lines 4-9). In each iteration, the algorithm generates a size-k seed set S^* by invoking the greedy algorithm on $\mathcal{R}_1$ (Line 5), after which it derives $\sigma^l(S^*)$ and $\hat{\sigma}^u(S^o)$ from $\mathcal{R}_2$ and $\mathcal{R}_1$, respectively, setting $\delta_1 = \delta_2 = \delta/(3i_{\max})$ (Line 6). Then, it computes $\alpha \leftarrow \sigma^l(S^*)/\hat{\sigma}^u(S^o)$ as the approximation guarantee of S^* (Line 7). By (5), (13), (14), and Lemma 4.2, α is correct with at least $1 - 2\delta/(3i_{\max})$ probability. If $\alpha \geq 1 - 1/e - \varepsilon$, the algorithm returns S^* and terminates; otherwise, it inserts new random RR sets into $\mathcal{R}_1$ and $\mathcal{R}_2$ to double their sizes, and then proceeds to the next iteration (Lines 8 and 9). In addition, in the $i_{\max}$-th iteration, it directly returns S^* regardless of the value of α.

The reason that OPIM-C ensures $1 - 1/e - \varepsilon$ approximation with at least $1 - \delta$ probability can be explained as follows. First, the algorithm has at most $i_{\max}$ iterations. In each of the first $i_{\max} - 1$ iterations, it produces a seed set S^* and an approximation guarantee α of S^* that is incorrect with at most $2\delta/(3i_{\max})$ probability, and it returns S^* if $\alpha \geq 1 - 1/e - \varepsilon$. By the union bound, it has at most $\frac{2}{3}$ probability to output an incorrect solution in the first $i_{\max} - 1$ iterations. Meanwhile, in the last iteration, it returns a seed set S^* generated by applying the greedy algorithm on $\mathcal{R}_1$, with $|\mathcal{R}_1| \geq \theta_{\max}$. By Lemma 6.1, this ensures that S^* is an $(1 - 1/e - \varepsilon)$-approximation with at least $1 - \delta/3$ probability. Therefore, the probability that OPIM-C returns an incorrect solution in any iteration is at most δ.

Time Complexity. The computation overhead of OPIM-C is incurred by (i) the generation of RR sets, (ii) the execution of the greedy algorithm, and (iii) the computation of $\sigma^l(S^*)$ and $\hat{\sigma}^u(S^o)$. To analyze the time complexity of OPIM-C, we first quantify the expected number of RR sets generated by OPIM-C as follows.

Lemma 6.2. *When $\delta \leq 1/2$, OPIM-C generates an expected number of $O\big((k \ln n + \ln(1/\delta))n\varepsilon^{-2}/\sigma(S^o)\big)$ RR sets.*

Previous work [39] shows that, under the *triggering model* [20] (which generalizes both the IC and LT models), the expected time required to generate a random RR set is $O(\frac{m}{n}\mathbb{E}[\sigma(\{v^*\})])$, where v^* is a node selected randomly from those in G with probabilities proportional to their in-degrees. In addition, $\mathbb{E}[\sigma(\{v^*\})] \leq \sigma(S^o)$. Combining this with Lemma 6.2 and Wald's equation [41], we can show that OPIM-C requires $O\big((k \ln n + \ln(1/\delta))(n+m)\varepsilon^{-2}\big)$ expected time in RR set generation.

By the analysis in Section 5, given two sets $\mathcal{R}_1$ and $\mathcal{R}_2$ of RR sets, OPIM-C requires $O(kn + \sum_{R \in \mathcal{R}_1 \cup \mathcal{R}_2} |R|)$ time to execute the greedy algorithm and compute $\sigma^l(S^*)$ and $\hat{\sigma}^u(S^o)$. The expected value of $|R|$ can be derived as follows.

Lemma 6.3. *$\mathbb{E}[|R|] = \mathbb{E}[\sigma(\{v\})] \leq \sigma(S^o)$, where v is a node selected uniformly at random from G.*

Based on Lemmas 6.2 and 6.3, we have the following result on the expected time complexity of OPIM-C.

Theorem 6.4. *When $\delta \leq 1/2$, OPIM-C runs in $O\big((k \ln n + \ln(1/\delta))(n+m)\varepsilon^{-2}\big)$ expected time under the triggering model.*

Proof. According to Wald's equation [41], OPIM-C requires $O(kn + \mathbb{E}[|\mathcal{R}_1 \cup \mathcal{R}_2|] \cdot \mathbb{E}[|R|])$ expected time to execute the greedy algorithm and compute $\sigma^l(S^*)$ and $\hat{\sigma}^u(S^o)$ in the last iteration after which OPIM-C stops. Recall that OPIM-C doubles the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ in each iteration. Thus, the total number of RR sets examined in all the iterations is within twice of that in the last iteration. On the other hand, there are at most $i_{\max}$ iterations such that

$$i_{\max} \leq \log_2(\frac{n}{\varepsilon^2 k}) + 1 \leq \log_2(\frac{n}{\varepsilon^2}) + 1 \leq \log_2(n) + \frac{2}{\varepsilon} + 1,$$

for any $0 < \varepsilon < 1$. Combining this with Lemmas 6.2 and 6.3, the total expected time used for executing the greedy algorithm and computing $\sigma^l(S^*)$ and $\hat{\sigma}^u(S^o)$ in all the iterations is

$$\begin{aligned} &O(kn \cdot i_{\max} + 2\mathbb{E}[|\mathcal{R}_1 \cup \mathcal{R}_2|] \cdot \mathbb{E}[|R|]) \\ &= O\big(kn(\ln n + 1/\varepsilon) + (k \ln n + \ln(1/\delta))n\varepsilon^{-2}\big) \\ &= O\big((k \ln n + \ln(1/\delta))n\varepsilon^{-2}\big). \end{aligned}$$

Combing this with $O\big((k \ln n + \ln(1/\delta))(n+m)\varepsilon^{-2}\big)$ expected time in RR set generation, OPIM-C runs in $O\big((k \ln n + \ln(1/\delta))(n+m)\varepsilon^{-2}\big)$ expected time. □

Comparison with IMM. OPIM-C and IMM have the same approximation guarantees and expected time complexity, but OPIM-C runs much faster in practice since it requires a much smaller number of RR sets to be generated. Specifically, IMM uses the same set $\mathcal{R}$ of RR sets for constructing the seed set S^* and deriving the lower bound of its expected spread. To bound S^*'s expected spread from below using $\mathcal{R}$, IMM uses a union bound of all size-k node sets to ensure that the concentration bound in Lemma 4.1 can be applied to the seed set S^* obtained from $\mathcal{R}$ itself. This will increase the failure probability by a factor of $\binom{n}{k}$. However, OPIM-C divides the RR sets into two disjoint groups $\mathcal{R}_1$ and $\mathcal{R}_2$. The solution S^* is obtained from $\mathcal{R}_1$ and the lower bound $\sigma^l(S^*)$ of $\sigma(S^*)$ is computed from $\mathcal{R}_2$. Since S^* is independent of the RR sets in $\mathcal{R}_2$, the concentration

bound in Lemma 4.1 can be applied directly without using a union bound. Moreover, the improved upper bound $\hat{\sigma}^u(S^o)$ of the optimal seed set S^o's expected spread in (13) can further speed up the OPIM-C algorithm. As a consequence, OPIM-C's practical efficiency is much higher to that of IMM, which shall be shown in Section 8.

7 RELATED WORK

Domingos et al. [10] are the first to study viral marketing from an algorithmic perspective. After that, Kempe et al. [20] formulate the influence maximization problem, and propose a greedy algorithm that returns $(1 - 1/e - \varepsilon)$-approximation for several influence diffusion models, by utilizing Monte Carlo simulations. Subsequently, there has been a large body of research on improved algorithms for influence maximization [1, 2, 4–9, 11–15, 18, 19, 22, 24, 28, 30, 32, 34, 35, 38, 39, 44, 45]. In particular, there is a long line of work [4–9, 11, 12, 15, 19, 22, 30, 32, 34, 35, 44, 45] that aims to develop heuristics to reduce the computation overhead of influence maximization at the cost of foregoing worst-case guarantees. In contrast, more recent work [2, 18, 28, 38, 39] focuses on algorithms that ensure $(1 - 1/e - \varepsilon)$-approximations with reduced computation overheads, by utilizing the reverse influence sampling technique proposed by Borgs et al. [2].

Specifically, Tang et al. [39] propose TIM, which is an RIS-based influence maximization algorithm runs in $O((k+\ell)(n+m)\log n/\varepsilon^2)$ expected time under the IC and LT models, and returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1 - n^{\ell}$ probability. They show that TIM outperforms Borgs et al.'s influence maximization method [2] in terms of both time complexity and empirical efficiency. After that, they develop a further improved algorithm, IMM [38], which retains TIM's theoretical guarantees but considerably reduces its computation overhead in practice. Subsequently, Nguyen et al. [28] present two RIS-based algorithms for influence maximization, SSA and D-SSA, and claim that they provide $(1-1/e-\varepsilon)$-approximations while incurring significantly smaller running time. However, Huang et al. [18] show that the theoretical analysis in [28] contains serious loopholes that invalidate the theoretical claims made, and that the experimental results have numerous anomalies. They also present a revised version of SSA, referred to as SSA-Fix, that restores its $(1 - 1/e - \varepsilon)$-approximation guarantee. Meanwhile, Nguyen et al. [29] also present a significantly modified version of D-SSA, referred to as D-SSA-Fix, that provides $(1 - 1/e - \varepsilon)$-approximations. However, the efficiency guarantee of D-SSA-Fix still remains unclear, as pointed out in Huang et al. [18]. As we discuss in Section 3.3, although the above algorithms can be adopted for OPIM, they result in unsatisfactory solutions due to the fundamental difference between OPIM and conventional influence maximization. Interested readers are also referred to Appendix C on a detailed analysis on the difference between our OPIM approach and D-SSA-Fix.

Finally, there is a series of recent work [3, 23, 26, 33, 36, 37, 43] that investigates other variants of influence maximization. Specifically, Lei et al. [23] propose to learn influence probabilities and maximize influence spread simultaneously on the input data. Chen et al. [3] study influence maximization by taking into account the users' topics of interests, and propose an index structure that offers heuristic solutions to online queries with topic information. Li et al. [26] investigate a keyword-based influence maximization problem that aims to identify a seed set that maximizes the expected spread over users whose keywords are relevant to a given advertisement. Wang et al. [43] consider influence maximization under a dynamic model where the diffusion probabilities in the social network are changing over time. Tang et al. [33, 36, 37] aim to optimize a profit metric that naturally combines the benefit and cost of influence spread. Techniques developed for these problems are inapplicable to the OPIM problem, due to the differences in problem definitions.

Table 2: Datasets.

Dataset	n	m	Type	Avg. degree
Pokec	1.6M	30.6M	directed	37.5
Orkut	3.1M	117.2M	undirected	76.3
LiveJournal	4.8M	69.0M	directed	28.5
Twitter	41.7M	1.5G	directed	70.5

8 EXPERIMENTS

This section experimentally evaluates our OPIM algorithms and our OPIM-C method against the state-of-the-art solutions. All experiments are carried out on a machine with an Intel Xeon 2.4GHz CPU and 64GB memory.

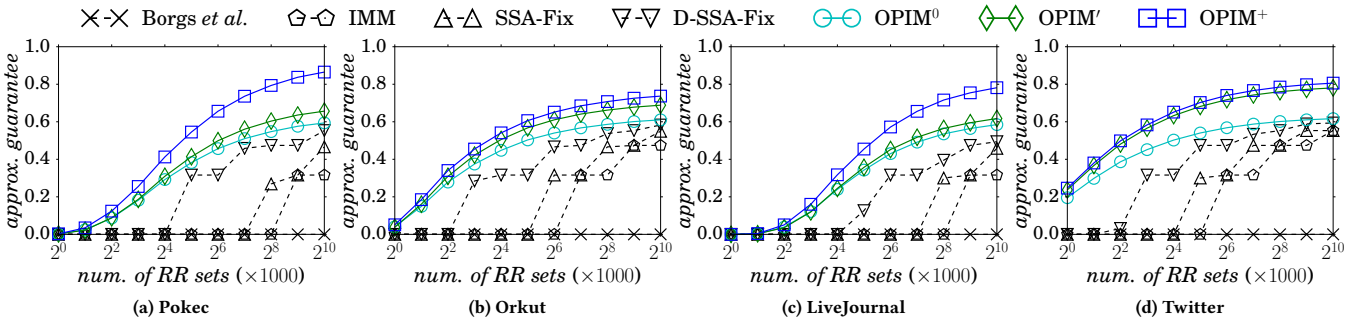
8.1 Experimental Setup

Datasets. We experiment with four real datasets with millions of nodes, namely, Pokec [25], Orkut [25], LiveJournal [25], and Twitter [21]. Table 2 shows the details of each dataset.

Algorithms. We evaluate seven algorithms for OPIM, namely, Borgs et al.'s OPIM algorithm [2], three versions of our OPIM method described in Sections 4 and 5, as well as the OPIM-adoptions of three state-of-the-art influence maximization algorithms that offers $(1 - 1/e - \varepsilon)$-approximations, namely, IMM [38], SSA-Fix [18], and D-SSA-Fix [29]. For our OPIM method, we use OPIM^0 to denote the vanilla version described in Section 4, OPIM$^+$ to denote the improved version that utilizes the upper bound $\hat{\sigma}^u(S^o)$, and OPIM$'$ to denote the version using an alternative upper bound $\sigma^\diamond(S^o)$ motivated by [24]. For IMM, SSA-Fix, and D-SSA-Fix, their OPIM-adoptions are as described in Section 3.3.

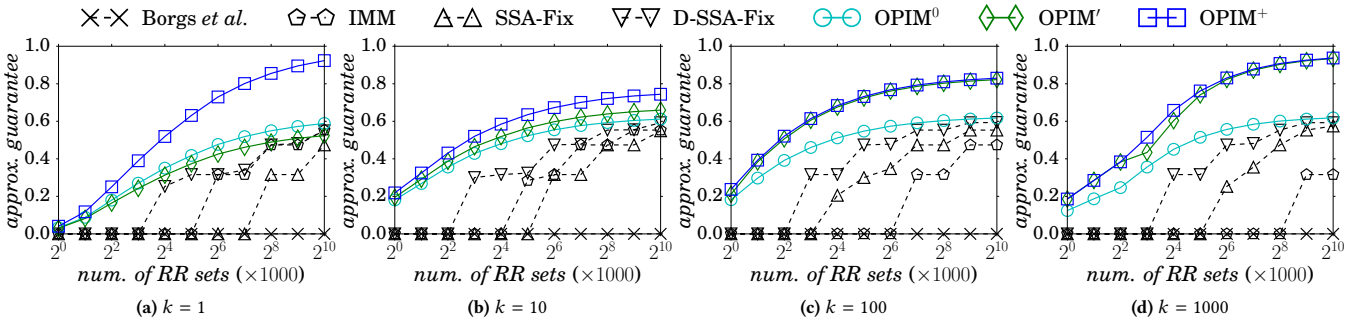
As for the conventional influence maximization problem, we evaluate OPIM-C (i.e., Algorithm 2) against IMM, SSA-Fix, and D-SSA-Fix. We adopt the C++ implementations of IMM, SSA-Fix, and D-SSA-Fix provided by their respective inventors, and we implement all other algorithms in C++.

Parameter Settings. For both the LT and IC models, we set the propagation probability $p(u, v)$ of each edge $\langle u, v\rangle$ to the inverse of v's in-degree, which is a setting frequently adopted by existing studies [4, 6, 18, 20, 28, 38, 39]. By default, we set the seed set size $k = 50$ and the failure probability $\delta = 1/n$. For OPIM, we report the approximation guarantee achieved by each algorithm when the number of RR sets generated reaches $2^i \times 1000$, with $i = 0, 1, 2, \ldots, 10$. (Note that all algorithms that we considered are based on RR sets, and that the generation of RR sets incurs the major overhead of each algorithm.) For each parameter setting, we repeat each algorithm 50 times and report the average measurement.



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

Figure 2: Approximation guarantee on various graphs under the LT model ($k = 50$).



(a) $k = 1$ (b) $k = 10$ (c) $k = 100$ (d) $k = 1000$

Figure 3: Approximation guarantee for different seed set sizes k on the Twitter dataset under the LT model.

In addition, for conventional influence maximization, we estimate the expected influence spread of the seed set returned by each algorithm by taking the average of its spreads over 10,000 Monte Carlo simulations.

8.2 OPIM Results under the LT Model

In the first set of experiments, we consider OPIM under the LT model. Figure 2 shows the approximation guarantees reported by different algorithms when $k = 50$ and the number of RR sets varies. Observe that Borgs et al.'s algorithm reports an approximation guarantee that is close to 0 in all cases. This is because the quality assurance derived by Borgs et al.'s algorithm is extremely loose, as discussed in Section 3.2. Meanwhile, all three versions of our OPIM algorithms report much better approximation guarantees than those by the OPIM-adoptions of IMM, SSA-Fix, and D-SSA-Fix. This, as analyzed in Section 3.3, is caused by the fact that the OPIM-adoptions cannot fully utilize the RR sets generated. Furthermore, OPIM$^+$ considerably outperforms OPIM^0 and OPIM$'$, since it reports an approximation ratio $\frac{\sigma^l(S^*)}{\hat{\sigma}^u(S^o)}$ that is much tighter than those by OPIM^0 and OPIM$'$. In addition, the approximation ratios reported by IMM, SSA-Fix, and D-SSA-Fix never exceed $1 - 1/e$, whereas those by our OPIM algorithms can be as large as 0.9. The reason is that our OPIM algorithms provide instance-specific approximation guarantees, whereas the OPIM-adoptions of IMM, SSA-Fix, and D-SSA-Fix can only offer approximation ratios over the worst-possible inputs.

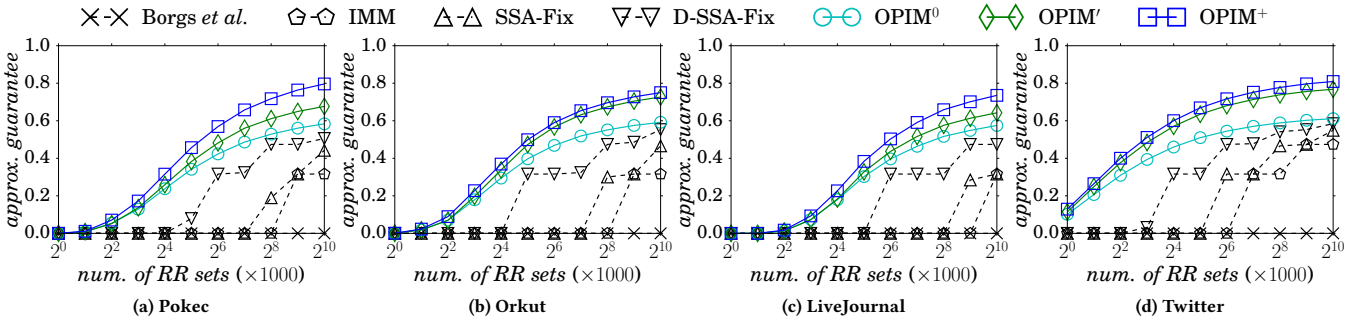
Figure 3 shows the approximation guarantee reported by each algorithm varying with the number of RR sets generated on the largest dataset, Twitter, for $k = 1, 10, 100, 1000$. Again, OPIM$^+$ consistently outperforms OPIM^0, which in turn is superior to the OPIM-adoptions of IMM, SSA-Fix, and D-SSA-Fix. Interestingly, OPIM$'$ outperforms OPIM^0 for $k = 10, 100, 1000$, but is inferior to the OPIM^0 when $k = 1$. The reason is that, as we analyze in Section 5, the upper bound $\sigma^\diamond(S^o)$ used by OPIM$'$ could be even looser than the upper bound $\sigma^u(S^o)$ used by OPIM^0. In contrast, the upper bound $\hat{\sigma}^u(S^o)$ used by OPIM$^+$ is never worse than $\sigma^u(S^o)$, as shown in Lemma 5.1.

8.3 OPIM Results under the IC Model

In our second set of experiments, we consider OPIM under the IC model. Figures 4 and 5 show the results. Similar to the case of the LT model, the results show that (i) our OPIM methods provide significantly higher approximation guarantees than the Borgs et al.'s algorithm as well as the OPIM-adoptions of IMM, SSA-Fix, and D-SSA-Fix, and (ii) due to our optimization techniques in Section 5, OPIM$^+$ outperforms OPIM and OPIM$'$ considerably. These results demonstrate the robustness of our OPIM solutions with respect to different information diffusion models.

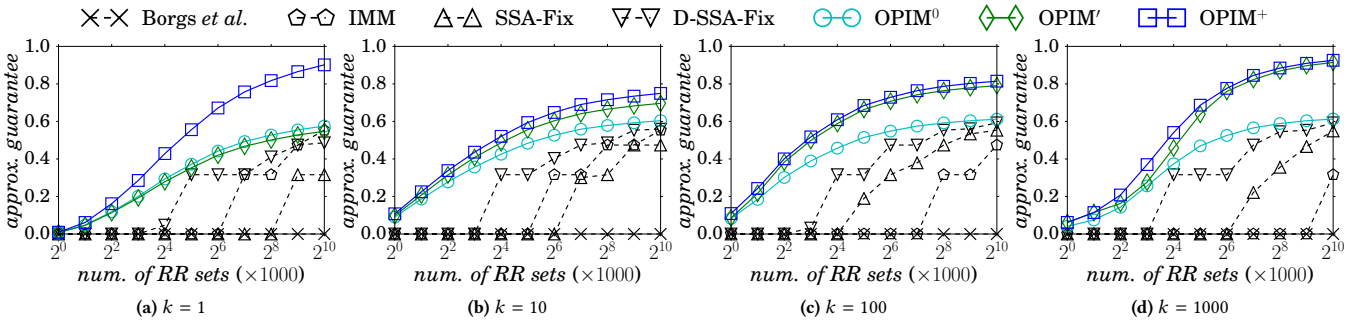
8.4 Conventional Influence Maximization

In our last set of experiments, we evaluate OPIM-C, i.e., the extension of our OPIM algorithms to address conventional influence maximization. We consider three versions of OPIM-C, namely, OPIM-C^0, OPIM-C$'$, and OPIM-C$^+$, which are based on OPIM^0, OPIM$'$, and OPIM$^+$, respectively. We compare them with IMM, SSA-Fix, and D-SSA-Fix when they return $(1 - 1/e - \varepsilon)$-approximation solutions



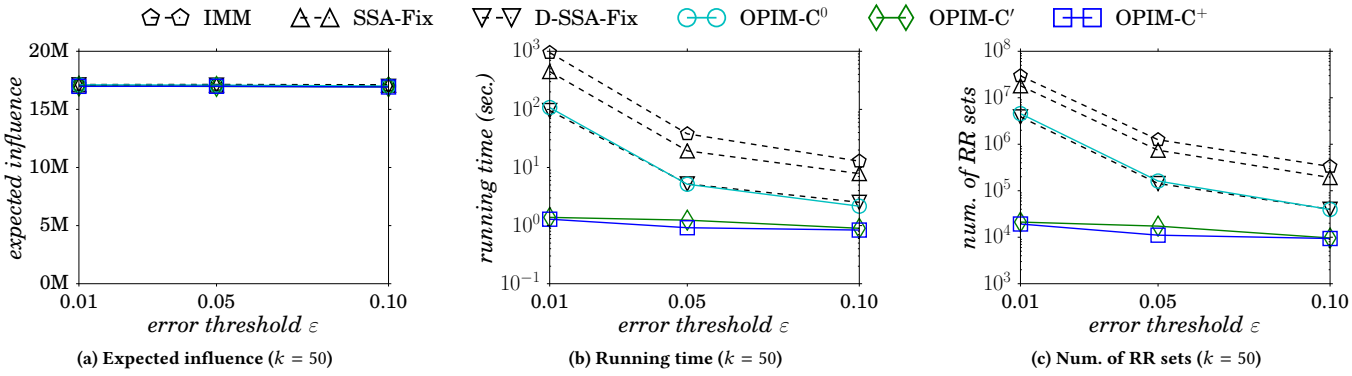
(a) Pokec (b) Orkut (c) LiveJournal (d) Twitter

Figure 4: Approximation guarantee on various graphs under the IC model ($k = 50$).



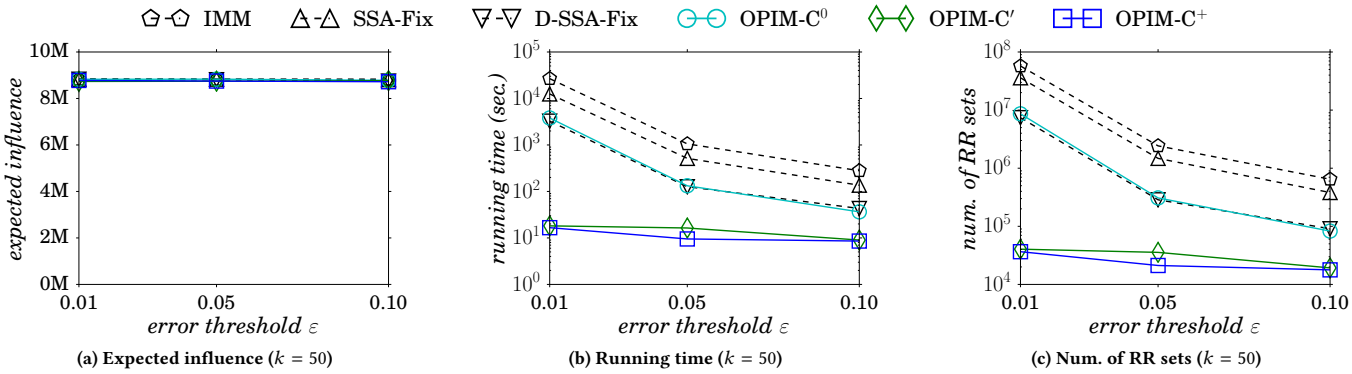
(a) $k = 1$ (b) $k = 10$ (c) $k = 100$ (d) $k = 1000$

Figure 5: Approximation guarantee for different seed set sizes k on the Twitter dataset under the IC model.



(a) Expected influence ($k = 50$) (b) Running time ($k = 50$) (c) Num. of RR sets ($k = 50$)

Figure 6: Influence maximization with $(1 - 1/e - \varepsilon)$-approximation on the Twitter dataset under the LT model.



(a) Expected influence ($k = 50$) (b) Running time ($k = 50$) (c) Num. of RR sets ($k = 50$)

Figure 7: Influence maximization with $(1 - 1/e - \varepsilon)$-approximation on the Twitter dataset under the IC model.

to influence maximization, setting $k = 50$ and varying ϵ from 0.01 to 0.1.

Figures 6 and 7 show the results from Twitter (i.e., the largest dataset tested) under the LT and IC models, respectively. In general, all algorithms tested yield similar expected spreads, as shown in Figures 6(a) and 7(a). From Figures 6(b) and 7(b), we can observe that OPIM-C^0 provides similar efficiency to the best existing method for influence maximization, which demonstrates the effectiveness of our technique. Furthermore, OPIM-C$^+$ significantly outperforms OPIM-C^0, as it utilizes a much tighter bound on approximation guarantee than OPIM-C^0 does. In particular, for the case of $\epsilon = 0.01$, OPIM-C$^+$ is up to three orders of magnitude faster than OPIM-C^0, IMM, SSA-Fix, and D-SSA-Fix. Meanwhile, OPIM-C$'$ is as efficient as OPIM-C$^+$ when $\epsilon = 0.01$ or $\epsilon = 0.1$, but is outperformed by OPIM-C$^+$ when $\epsilon = 0.05$. This demonstrates that the instance-specific approximation ratio derived by OPIM-C$'$ is not as robust as OPIM-C$^+$.

9 CONCLUSION

This paper studies the problem of online processing of influence maximization (OPIM), and presents algorithms that provide both rigorous theoretical guarantees and superior empirical effectiveness. The core of our algorithms is a set of techniques for deriving instance-specific approximation guarantees of a given seed set. We show that these techniques not only enable us to address the deficiency of existing OPIM solutions, but also allow us to extend our OPIM methods to tackle conventional influence maximization. With extensive experiments on real data, we show that our solutions considerably outperform the state of the art for both OPIM and conventional influence maximization. For future work, we plan to extend our OPIM algorithms to handle other variants of influence maximization, and to develop online processing algorithms for other graph problems.

ACKNOWLEDGMENTS

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 1 under Grant 2017-T1-002-024 and Tier 2 under Grant MOE2015-T2-2-114. Xiaokui Xiao was partially supported by MOE, Singapore under Grant MOE2015-T2-2-069, and by NUS, Singapore, under an SUG.

REFERENCES

[1] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. 2017. Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study. In *Proc. ACM SIGMOD*. 651–666.
[2] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *Proc. SODA*. 946–957.
[3] Shuo Chen, Ju Fan, Guoliang Li, Jianhua Feng, Kian-Lee Tan, and Jinhui Tang. 2015. Online Topic-Aware Influence Maximization. *Proc. VLDB Endowment* 8, 6 (2015), 666–677.
[4] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable Influence Maximization for Prevalent Viral Marketing in Large-scale Social Networks. In *Proc. ACM KDD*. 1029–1038.
[5] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient Influence Maximization in Social Networks. In *Proc. ACM KDD*. 199–208.
[6] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable Influence Maximization in Social Networks Under the Linear Threshold Model. In *Proc. IEEE ICDM*. 88–97.
[7] Suqi Cheng, Huawei Shen, Junming Huang, Wei Chen, and Xueqi Cheng. 2014. IMRank: Influence Maximization via Finding Self-consistent Ranking. In *Proc. ACM SIGIR*. 475–484.
[8] Suqi Cheng, Huawei Shen, Junming Huang, Guoqing Zhang, and Xueqi Cheng. 2013. StaticGreedy: Solving the Scalability-accuracy Dilemma in Influence Maximization. In *Proc. ACM CIKM*. 509–518.
[9] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. 2014. Sketch-Based Influence Maximization and Computation: Scaling Up with Guarantees. In *Proc. ACM CIKM*. 629–638.
[10] Pedro Domingos and Matt Richardson. 2001. Mining the Network Value of Customers. In *Proc. ACM KDD*. 57–66.
[11] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic Influence Maximization: Combining Scalability and Efficiency with Opinion-Aware Models. In *Proc. ACM SIGMOD*. 743–758.
[12] Sainyam Galhotra, Akhil Arora, Srinivas Virinchi, and Shourya Roy. 2015. ASIM: A Scalable Algorithm for Influence Maximization Under the Independent Cascade Model. In *Proc. WWW Companion*. 35–36.
[13] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2011. A Data-based Approach to Social Influence Maximization. *Proc. VLDB Endowment* 5, 1 (2011), 73–84.
[14] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. 2011. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *Proc. WWW Companion*. 47–48.
[15] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. 2011. SIMPATH: An Efficient Algorithm for Influence Maximization Under the Linear Threshold Model. In *Proc. IEEE ICDM*. 211–220.
[16] Peter J. Haas and Joseph M. Hellerstein. 1999. Ripple Joins for Online Aggregation. In *Proc. SIGMOD*. 287–298.
[17] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. 1997. Online Aggregation. In *Proc. SIGMOD*. 171–182.
[18] Keke Huang, Sibo Wang, Glenn Bevilacqua, Xiaokui Xiao, and Laks V. S. Lakshmanan. 2017. Revisiting the Stop-and-Stare Algorithms for Influence Maximization. *Proc. VLDB Endowment* 10, 9 (2017), 913–924.
[19] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. IRIE: Scalable and Robust Influence Maximization in Social Networks. In *Proc. IEEE ICDM*. 918–923.
[20] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the Spread of Influence Through a Social Network. In *Proc. ACM KDD*. 137–146.
[21] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a Social Network or a News Media?. In *Proc. WWW*. 591–600.
[22] Jong Ryul Lee and Chin Wan Chung. 2014. A Fast Approximation for Influence Maximization in Large Social Networks. In *WWW Companion*. 1157–1162.
[23] Siyu Lei, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart. 2015. Online Influence Maximization. In *Proc. ACM KDD*. 645–654.
[24] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective Outbreak Detection in Networks. In *Proc. ACM KDD*. 420–429.
[25] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (2014).
[26] Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. 2015. Real-time Targeted Influence Maximization for Online Advertisements. *Proc. VLDB Endowment* 8, 10 (2015), 1070–1081.
[27] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An Analysis of Approximations for Maximizing Submodular Set Functions-I. *Mathematical Programming* 14, 1 (1978), 265–294.
[28] 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 *Proc. ACM SIGMOD*. 695–710.
[29] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 22 Feb 2017. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-Scale Networks. arXiv preprint, https://arxiv.org/abs/1605.07990v3. (22 Feb 2017).
[30] 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 *Proc. AAAI*. 138–144.
[31] Jeff John Roberts. 2016. Facebook and Google Are Big Winners as Political Ad Money Moves Online. *Fortune* (2016).
[32] Guojie Song, Xiabing Zhou, Yu Wang, and Kunqing Xie. 2015. Influence Maximization on Large-Scale Mobile Social Network: A Divide-and-Conquer Method. *IEEE Transactions on Parallel and Distributed Systems* 26, 5 (2015), 1379–1392.
[33] Jing Tang, Xueyan Tang, and Junsong Yuan. 2016. Profit Maximization for Viral Marketing in Online Social Networks. In *Proc. IEEE ICNP*. 1–10.
[34] Jing Tang, Xueyan Tang, and Junsong Yuan. 2017. Influence Maximization Meets Efficiency and Effectiveness: A Hop-Based Approach. In *Proc. IEEE/ACM ASONAM*. 64–71.
[35] Jing Tang, Xueyan Tang, and Junsong Yuan. 2018. An Efficient and Effective Hop-Based Approach for Inluence Maximization in Social Networks. *Social Network Analysis and Mining* 8, 10 (2018).
[36] Jing Tang, Xueyan Tang, and Junsong Yuan. 2018. Profit Maximization for Viral Marketing in Online Social Networks: Algorithms and Analysis. *IEEE Transactions on Knowledge and Data Engineering* (2018).
[37] Jing Tang, Xueyan Tang, and Junsong Yuan. 2018. Towards Profit Maximization for Online Social Network Providers. In *Proc. IEEE INFOCOM*. 1178–1186.

[38] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *Proc. ACM SIGMOD*. 1539–1554.
[39] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence Maximization: Near-optimal Time Complexity Meets Practical Efficiency. In *Proc. ACM SIGMOD*. 75–86.
[40] Vijay V. Vazirani. 2003. *Approximation Algorithms*. Springer.
[41] Abraham Wald. 1947. *Sequential Analysis*. Wiley.
[42] Alastair J. Walker. 1977. An Efficient Method for Generating Discrete Random Variables with General Distributions. *ACM Trans. Math. Software* 3, 3 (1977), 253–256.
[43] Yanhao Wang, Qi Fan, Yuchen Li, and Kian-Lee Tan. 2017. Real-time Influence Maximization on Dynamic Social Streams. *Proc. VLDB Endowment* 10, 7 (2017), 805–816.
[44] Chuan Zhou, Peng Zhang, Jing Guo, and Li Guo. 2014. An Upper Bound Based Greedy Algorithm for Mining Top-k Influential Nodes in Social Networks. In *Proc. WWW Companion*. 421–422.
[45] Chuan Zhou, Peng Zhang, Jing Guo, Xingquan Zhu, and Li Guo. 2013. UBLF: An Upper Bound Based Approach to Discover Influential Nodes in Social Networks. In *Proc. IEEE ICDM*. 907–916.

A GENERATION OF RR SETS

Under the LT model, a random RR set R on G can be generated in three steps:

(1) Select a node v uniformly at random from V.
(2) Generate a random walk from v that follows the incoming edges of each node. Specifically, at each step of the random walk, we examine the set I of in-neighbors of the current node u, and stop at u with probability $1-\sum_{w\in I} p(w,u)$. With the other $\sum_{w\in I} p(w,u)$ probability, we sample a node x from a distribution $f(x)$ on I with $f(x) \sim p(x,u)$, and walk to x.
(3) Insert all nodes in the random walk (including v) into R.

Similarly, under the IC model, a random RR set R on G can also be constructed in three steps:

(1) Select a node v uniformly at random from V.
(2) Perform a stochastic breadth first search (BFS) that starts from v and follows the incoming edges of each node. In particular, for each node u encountered during the BFS, we inspect the set I of incoming edges of u. For each edge $\langle w,u\rangle$ in I, we ignore it with $1-p(w,u)$ probability; with the other $p(w,u)$ probability, we allow the BFS to traverse to w from u (if w has not been traversed).
(3) Insert into R all nodes that are traversed during the stochastic BFS.

Previous work [39] shows that, under the triggering model (which generalizes both the LT and IC models), a random RR set can be constructed in $O(\frac{m}{n}\mathbb{E}[\sigma(\{v^*\})])$ expected time, where v^* is a randomly selected node with probability proportional to its in-degree in G. However, under the LT model, it only takes $O(1)$ time to sample (at most) one in-neighbor w of u according to the probability $p(w,u)$ using the alias method [42]. Thus, under the LT model, the expected time complexity for generating a RR set R is $O(|R|) = O(\mathbb{E}[\sigma(\{v\})])$ (see Lemma 6.3), where v is a node uniformly selected at random. Note that the alias method uses $O(n+m)$ preprocessing time.

B PROOFS

Proof of Lemma 4.4. $f(x)$ can be rewritten as

$$\begin{aligned} f(x) &= \Lambda_2(S^*) + \frac{2x}{3} - \sqrt{2x\Lambda_2(S^*) + \frac{4x^2}{9}} \\ &= \Lambda_2(S^*) + \frac{\frac{4x^2}{9} - 2x\Lambda_2(S^*) - \frac{4x^2}{9}}{\frac{2x}{3} + \sqrt{2x\Lambda_2(S^*) + \frac{4x^2}{9}}} \\ &= \Lambda_2(S^*) - \frac{2\Lambda_2(S^*)}{\frac{2}{3} + \sqrt{\frac{2\Lambda_2(S^*)}{x} + \frac{4}{9}}}. \end{aligned}$$

Thus, $f(x)$ is decreasing with x. Meanwhile, $g(x)$ is increasing with x. Therefore, for any $\delta_1, \delta_2 \in (0,\delta)$ satisfying $\delta_1+\delta_2 \le \delta$,

$$\frac{\sigma^l(S^*)}{\sigma^u(S^o)} = \frac{f(\ln\frac{1}{\delta_1})}{g(\ln\frac{1}{\delta_2})}\cdot\frac{\theta_1}{\theta_2} \le \frac{f(\ln\frac{1}{\delta})}{g(\ln\frac{1}{\delta})}\cdot\frac{\theta_1}{\theta_2}. \tag{18}$$

On the other hand, when we set $\delta_1=\delta_2=\delta/2$,

$$\alpha = \frac{\sigma^l(S^*)}{\sigma^u(S^*)} = \frac{f(\ln\frac{2}{\delta})}{g(\ln\frac{2}{\delta})}\cdot\frac{\theta_1}{\theta_2}. \tag{19}$$

Thus, by (18) and (19), we have

$$\frac{\alpha}{\alpha'} \ge \frac{f(\ln\frac{2}{\delta})\cdot g(\ln\frac{1}{\delta})}{f(\ln\frac{1}{\delta})\cdot g(\ln\frac{2}{\delta})}.$$

Hence, the lemma is proved. □

Proof of Lemma 6.2. Let

$$\varepsilon_1=\varepsilon,\quad \tilde{\varepsilon}_1=\varepsilon-(1-1/e)\varepsilon_1=\varepsilon/e,\quad \hat{\varepsilon}_1=\sqrt{\frac{2a_1 n}{\sigma(S^o)\theta_1}},$$

$$\varepsilon_2=\sqrt{\frac{2a_1 n}{\sigma(S^*)\theta_2}},\quad \text{and } \tilde{\varepsilon}_2=\left(\sqrt{\frac{2a_1\sigma(S^*)\theta_2}{n}+\frac{a_1^2}{9}}+\frac{a_1}{3}\right)\cdot\frac{n}{\sigma(S^*)\theta_2},$$

where $a_1 = c\ln(\frac{3i_{\max}}{\delta})$ for any $c\ge 1$. Furthermore, let

$$\theta' = \max\left\{\frac{2n\ln\frac{6}{\delta}}{\varepsilon_1^2\sigma(S^o)}, \frac{(2+2\tilde{\varepsilon}_1/3)n\ln\frac{6\binom{n}{k}}{\delta}}{\tilde{\varepsilon}_1^2\sigma(S^o)}, \frac{27n\ln\frac{3i_{\max}}{\delta}}{(1-1/e-\varepsilon)\varepsilon_1^2\sigma(S^o)}\right\}.$$

It is easy to verify that $\theta' = O\big((k\ln n+\ln(1/\delta))n\varepsilon^{-2}/\sigma(S^o)\big)$. [1]

Then, when $\theta_1=\theta_2=c\theta'$, according to Lemma 4.1, we have the following results.

$$\Pr\left[\Lambda_1(S^o)\cdot\frac{n}{\theta_1} < (1-\varepsilon_1)\cdot\sigma(S^o)\right] \le \left(\frac{\delta}{6}\right)^c, \tag{20}$$

$$\Pr\left[\Lambda_1(S^*)\cdot\frac{n}{\theta_1} > \sigma(S^*)+\tilde{\varepsilon}_1\cdot\sigma(S^o)\right] \le \left(\frac{\delta}{6\binom{n}{k}}\right)^c, \tag{21}$$

$$\Pr\left[\Lambda_1(S^o)\cdot\frac{n}{\theta_1} < (1-\hat{\varepsilon}_1)\cdot\sigma(S^o)\right] \le \left(\frac{\delta}{3i_{\max}}\right)^c, \tag{22}$$

$$\Pr\left[\Lambda_2(S^*)\cdot\frac{n}{\theta_2} < (1-\varepsilon_2)\cdot\sigma(S^*)\right] \le \left(\frac{\delta}{3i_{\max}}\right)^c, \tag{23}$$

$$\Pr\left[\Lambda_2(S^*)\cdot\frac{n}{\theta_2} > (1+\tilde{\varepsilon}_2)\cdot\sigma(S^*)\right] \le \left(\frac{\delta}{3i_{\max}}\right)^c. \tag{24}$$

[1] Without loss of generality, we assume $\varepsilon \le 0.5$. If $\varepsilon > 0.5$, OPIM-C achieves a higher approximation of $1-1/e-0.5$ with $O\big((k\ln n+\ln(1/\delta))n/\sigma(S^o)\big)$ RR sets generated.

In particular, (20) is obtained by

$$\text{l.h.s.} \le \exp\Big(-\frac{\varepsilon_1^2\sigma(S^o)\frac{\theta_1}{n}}{2}\Big) \le \Big(\frac{\delta}{6}\Big)^c,$$

and (21) is obtained by

$$\text{l.h.s.} \le \exp\Big(-\frac{\tilde{\varepsilon}_1^2\sigma(S^o)\frac{\theta_1}{n}}{2\frac{\sigma(S^*)}{\sigma(S^o)}+2\tilde{\varepsilon}_1/3}\Big) \le \exp\Big(-\frac{\sigma(S^o)\frac{\theta_1}{n}\tilde{\varepsilon}_1^2}{2+2\tilde{\varepsilon}_1/3}\Big) \le \Big(\frac{\delta}{6\binom{n}{k}}\Big)^c,$$

and (22)–(24) are obtained similarly.

Note that S^* is obtained from the set $\mathcal{R}_1$ of RR sets, which is not independent of $\mathcal{R}_1$. To fulfill the requirement of Lemma 4.1, we multiply the failure probability with a factor of $\binom{n}{k}$ to ensure all size-k node sets to be included. By the union bound, the probability that none of the events in (20)–(24) happens is at least

$$1-\left(\Big(\frac{\delta}{6}\Big)^c+\Big(\frac{\delta}{6\binom{n}{k}}\Big)^c\cdot\binom{n}{k}+3\cdot\Big(\frac{\delta}{3i_{\max}}\Big)^c\right) \ge 1-\delta^c.$$

On the other hand, it is easy to get that

$$\hat{\varepsilon}_1 \le \sqrt{\frac{2(1-1/e-\varepsilon)\varepsilon_1^2}{27}} < \varepsilon_1/3.$$

Meanwhile, when the events in (20) and (21) do not happen, we have

$$\begin{aligned}\sigma(S^*) &\ge \Lambda_1(S^*)\cdot\frac{n}{\theta_1}-\tilde{\varepsilon}_1\cdot\sigma(S^o)\\ &\ge (1-1/e)\Lambda_1(S^o)\cdot\frac{n}{\theta_1}-\tilde{\varepsilon}_1\cdot\sigma(S^o)\\ &\ge (1-1/e)(1-\varepsilon_1)\cdot\sigma(S^o)-\tilde{\varepsilon}_1\cdot\sigma(S^o)\\ &= (1-1/e-\varepsilon)\cdot\sigma(S^o). \end{aligned} \tag{25}$$

Thus, based on the definitions of ε_2 and $\tilde{\varepsilon}_2$, we have

$$\varepsilon_2 = \sqrt{\frac{2(1-1/e-\varepsilon)\sigma(S^o)\varepsilon_1^2}{27\sigma(S^*)}} < \varepsilon_1/3,$$

$$\text{and}\quad \tilde{\varepsilon}_2 = \sqrt{\frac{a_1(2+2\tilde{\varepsilon}_2/3)n}{\sigma(S^*)\theta_2}} \le \sqrt{\frac{(2+2\tilde{\varepsilon}_2/3)\varepsilon_1^2}{27}} < \varepsilon_1/3.$$

In addition, when the event in (22) does not happen, we have

$$\Big(\sqrt{\Lambda_1(S^o)+\frac{a_1}{2}}+\sqrt{\frac{a_1}{2}}\Big)^2\cdot\frac{n}{\theta_1} \ge \sigma(S^o).$$

Thus, it holds that

$$\begin{aligned}1-\hat{\varepsilon}_1 &= 1-\sqrt{\frac{2a_1 n}{\sigma(S^o)\theta_1}}\\ &\le 1-\frac{\sqrt{2a_1}}{\sqrt{\Lambda_1(S^o)+\frac{a_1}{2}}+\sqrt{\frac{a_1}{2}}}\\ &\le \frac{\Lambda_1^u(S^o)}{\Big(\sqrt{\Lambda_1^u(S^o)+\frac{a_1}{2}}+\sqrt{\frac{a_1}{2}}\Big)^2}.\end{aligned}$$

By setting $\delta_1 = \delta/(3i_{\max})$ such that $\ln(1/\delta_1) \le a_1$, it follows from (13) that

$$\hat{\sigma}^u(S^o) \le \Big(\sqrt{\Lambda_1^u(S^o)+\frac{a_1}{2}}+\sqrt{\frac{a_1}{2}}\Big)^2\cdot\frac{n}{\theta_1} \le \frac{\Lambda_1^u(S^o)}{1-\hat{\varepsilon}_1}\cdot\frac{n}{\theta_1}. \tag{26}$$

Similarly, when the event in (24) does not happen, we have

$$\Big(\sqrt{\Lambda_2(S^*)+\frac{2a_1}{9}}-\sqrt{\frac{a_1}{2}}\Big)^2-\frac{a_1}{18} \le \sigma(S^*)\cdot\frac{\theta_2}{n}.$$

Thus, it holds that

$$\begin{aligned}\Lambda_2(S^*)-\tilde{\varepsilon}_2\sigma(S^*)\cdot\frac{\theta_2}{n} &= \Lambda_2(S^*)-\Big(\sqrt{2a_1\sigma(S^*)\cdot\frac{\theta_2}{n}+\frac{a_1^2}{9}}+\frac{a_1}{3}\Big)\\ &\le \Lambda_2(S^*)-\Big(\sqrt{2a\Lambda_2(S^*)+\frac{4a_1^2}{9}}-a_1+\frac{a_1}{3}\Big)\\ &= \Big(\sqrt{\Lambda_2(S^*)+\frac{2a_1}{9}}-\sqrt{\frac{a_1}{2}}\Big)^2-\frac{a_1}{18}.\end{aligned}$$

By setting $\delta_2 = \delta/(3i_{\max})$ such that $\ln(1/\delta_2) \le a_1$, it follows from (5) that

$$\sigma^l(S^*) \ge \Lambda_2(S^*)\cdot\frac{n}{\theta_2}-\tilde{\varepsilon}_2\sigma(S^*). \tag{27}$$

Putting (26) and (27) together, when none of the events in (20)–(24) happens, we have

$$\begin{aligned}\frac{\sigma^l(S^*)}{\hat{\sigma}^u(S^o)} &\ge \frac{\Lambda_2(S^*)-\tilde{\varepsilon}_2\sigma(S^*)\cdot\frac{\theta_2}{n}}{\Lambda_1^u(S^o)/(1-\hat{\varepsilon}_1)}\\ &\ge \frac{(1-\hat{\varepsilon}_1)(1-\varepsilon_2-\tilde{\varepsilon}_2)\sigma(S^*)\cdot\frac{\theta_2}{n}}{\Lambda_1^u(S^o)}\\ &> (1-\hat{\varepsilon}_1-\varepsilon_2-\tilde{\varepsilon}_2)\frac{\sigma(S^*)\cdot\frac{\theta_2}{n}}{\Lambda_1(S^*)}\frac{\Lambda_1(S^*)}{\Lambda_1^u(S^o)}\\ &> (1-\varepsilon_1)\frac{\Lambda_1(S^*)-\tilde{\varepsilon}_1\cdot\sigma(S^o)\cdot\frac{\theta_2}{n}}{\Lambda_1(S^*)}(1-1/e)\\ &\ge (1-\varepsilon_1)\frac{\Lambda_1(S^*)-\tilde{\varepsilon}_1\Lambda_1(S^o)/(1-\varepsilon_1)}{\Lambda_1(S^*)}(1-1/e)\\ &\ge (1-\varepsilon_1)\Big(1-\frac{\tilde{\varepsilon}_1/(1-\varepsilon_1)}{1-1/e}\Big)(1-1/e)\\ &= 1-1/e-\varepsilon.\end{aligned} \tag{28}$$

Therefore, when $\theta_1 = \theta_2 = c\theta'$ RR sets are generated, OPIM-C does not stop only if at least one of the events in (20)–(24) happens. The probability is at most δ^c.

Let j be the first iteration that the number of RR sets generated by OPIM-C reaches θ'. From this iteration onward, the expected number of RR sets further generated is at most

$$\begin{aligned}2\cdot\sum_{z\ge j}\theta_0\cdot 2^z\cdot\delta^{2^{z-j}} &= 2\cdot 2^j\cdot\theta_0\sum_{z=0}2^z\cdot\delta^{2^z}\\ &\le 4\theta'\sum_{z=0}2^{-2^z+z}\\ &\le 4\theta'\sum_{z=0}2^{-z}\\ &\le 8\theta'.\end{aligned}$$

The first inequality is due to $2^{j-1}\cdot\theta_0 < \theta'$ and $\delta \le 1/2$, and the second inequality is due to $-2^z + z \le -z$. If the algorithm stops before this iteration, there are at most $2\theta'$ RR sets generated. Therefore, the expected number of RR sets generated is less than $10\theta'$, which is $O\big((k\ln n+\ln(1/\delta))\cdot n\cdot\varepsilon^{-2}/\sigma(S^o)\big)$.

Hence, the lemma is proved. □

Algorithm 3: D-SSA Algorithm

Input: Graph G, $0 \leq \varepsilon, \delta \leq 1$, and k
Output: An $(1-1/e-\varepsilon)$-optimal solution, S^*

1 $\theta'_{\max} = 8(1-1/e)\frac{\ln(6/\delta)+\ln\binom{n}{k}}{\varepsilon^2}\frac{n}{k}$;
2 $i'_{\max} = \lceil \log_2(\frac{2\theta'_{\max}\varepsilon^2}{(2+2\varepsilon/3)\ln(3/\delta)}) \rceil$; $i = 0$;
3 $\theta'_0 = \frac{(2+2\varepsilon/3)\ln(3i_{\max}/\delta)}{\varepsilon^2}$; $\Lambda_1 = 1+(1+\varepsilon)\frac{(2+2\varepsilon/3)\ln(3i_{\max}/\delta)}{\varepsilon^2}$;
4 **repeat**
5 $\quad i \leftarrow i+1$;
6 $\quad \mathcal{R}_1 = \{R_1, \ldots, R_{\theta'_0 \cdot 2^{i-1}}\}$ and $\theta_1 = |\mathcal{R}_1|$;
7 $\quad \mathcal{R}_2 = \{R_{\theta'_0 \cdot 2^{i-1}+1}, \ldots, R_{\theta'_0 \cdot 2^i}\}$ and $\theta_2 = |\mathcal{R}_2|$;
8 $\quad S^* \leftarrow \text{Greedy}(\mathcal{R}_1, k)$;
9 $\quad$ **if** $\Lambda_1(S^*) \geq \Lambda_1$ **then**
10 $\quad\quad \sigma_1(S^*) \leftarrow \Lambda_1(S^*) \cdot n/\theta_1$; $\sigma_2(S^*) \leftarrow \Lambda_2(S^*) \cdot n/\theta_2$;
11 $\quad\quad \varepsilon_a \leftarrow \sigma_1(S^*)/\sigma_2(S^*) - 1$;
12 $\quad\quad \varepsilon_b \leftarrow \varepsilon\sqrt{\frac{n(1+\varepsilon)}{2^{i-1}\sigma_2(S^*)}}$;
13 $\quad\quad \varepsilon_c \leftarrow \varepsilon\sqrt{\frac{n(1+\varepsilon)(1-1/e-\varepsilon)}{(1+\varepsilon/3)2^{i-1}\sigma_2(S^*)}}$;
14 $\quad\quad \varepsilon_i = (\varepsilon_a+\varepsilon_b+\varepsilon_a\varepsilon_b)(1-1/e-\varepsilon)+(1-1/e)\varepsilon_c$;
15 $\quad\quad$ **if** $\varepsilon_i \leq \varepsilon$ **then**
16 $\quad\quad\quad$ **return** S^*;
17 **until** $\theta_1 \geq \theta'_{\max}$;
18 **return** S^*;

Proof of Lemma 6.3. Let $\sigma(S, v)$ denote the probability that a node set S influences v. Thus, it is easy to get that

$$\sigma(S) = \sum_{v \in V} \sigma(S, v). \tag{29}$$

On the other hand, for each node $u \in V$, the probability that it appears in R equals the probability that a singleton seed set $\{u\}$ can activate v in an influence diffusion process. Thus,

$$\mathbb{E}[|R|] = \frac{1}{n}\sum_{v \in V}\sum_{u \in V} \sigma(\{u\}, v). \tag{30}$$

Therefore, (29) and (30) together yield

$$\begin{aligned}\mathbb{E}[|R|] &= \frac{1}{n}\sum_{v \in V}\sum_{u \in V} \sigma(\{u\}, v)\\ &= \frac{1}{n}\sum_{u \in V}\sum_{v \in V} \sigma(\{u\}, v)\\ &= \frac{1}{n}\sum_{u \in V} \sigma(\{u\})\\ &= \mathbb{E}[\sigma(\{u\})]\\ &\leq \sigma(S^o),\end{aligned}$$

where the inequality is due to the fact that $\sigma(S^o)$ is the maximum expected spread of any size-k node set. □

C NOTE ON D-SSA-FIX

Algorithm 3 presents the pseudo-code of D-SSA-Fix [29], using the notations defined in our paper. At the first glance, D-SSA-Fix may seem somewhat similar to our OPIM approach, since Line 14 of Algorithm 3 seems to provide the instance-specific approximation guarantee based on the RR sets generated by the algorithm. Unfortunately, D-SSA-Fix still requires the user to specify a predefined approximation error threshold ε, and it cannot be easily extended to handle OPIM, since the value of ε_i in Line 14 does not ensure that the seed set S^* returned is an $(1-1/e-\varepsilon_i)$-approximation, as we explained in the following.

For any required failure probability δ' (which is set to $\frac{\delta}{3i'_{\max}}$ in D-SSA-Fix), let $\hat{\varepsilon}$ be the unique root of

$$\hat{\varepsilon}^2 = \frac{(2+2\hat{\varepsilon}/3)n}{\theta_2\sigma(S^*)}\ln(1/\delta').$$

Then, by Chernoff bound in Lemma 4.1, if $\varepsilon_b \geq \hat{\varepsilon}$, we have

$$\Pr\left[\Lambda_2(S^*)\cdot\frac{n}{\theta_2} > (1+\varepsilon_b)\sigma(S^*)\right] \leq \delta'.$$

However, when $\varepsilon_b < \hat{\varepsilon}$, we observe that $\Lambda_2(S^*)\cdot\frac{n}{\theta_2} > (1+\varepsilon_b)\sigma(S^*)$ may hold with probability larger than δ'. We will show that $\varepsilon_b < \hat{\varepsilon}$ may happen with high probability, due to which D-SSA-Fix cannot provide any approximation guarantee for OPIM.

Meanwhile, by the definition of ε_b (Line 12), we have

$$\varepsilon_b^2 = \frac{n(1+\varepsilon)\varepsilon^2}{2^{i-1}\sigma_2(S^*)} = \frac{n(1+\varepsilon)\varepsilon^2\theta'_0}{\theta_2\sigma_2(S^*)} = \frac{n(1+\varepsilon)(2+2\varepsilon/3)}{\theta_2\sigma_2(S^*)}\ln(1/\delta'),$$

where θ'_0 is defined in Line 3 and $\sigma_2(S^*) = \Lambda_2(S^*)\cdot\frac{n}{\theta_2}$ (Line 10). Thus, we have

$$\frac{\varepsilon_b^2}{\hat{\varepsilon}^2} = \frac{(2+2\varepsilon/3)(1+\varepsilon)\sigma(S^*)}{(2+2\hat{\varepsilon}/3)\sigma_2(S^*)}.$$

Suppose that we are to maximize the expected spread for $k = 1$, $\delta' = 10^{-3}$, on a graph with $n = 10^5$ nodes and $m = 0$ edges. Thus, no matter which node is selected as the seed, its expected influence spread is always 1. Suppose that $\theta_2 = 10^5$ RR sets are generated. Then, we can compute that $\hat{\varepsilon} = 6.67$. Meanwhile,

$$\Pr[\Lambda_2(S^*) = 0] = (1-1/n)^{\theta_2} = 0.37.$$

Thus, with probability 0.63, $\sigma_2(S^*) \geq 1 = \sigma(S^*)$, which implies that even when setting $\varepsilon = 1-1/e$,

$$\frac{\varepsilon_b^2}{\hat{\varepsilon}^2} = \frac{(2+2\varepsilon/3)(1+\varepsilon)\sigma(S^*)}{(2+2\hat{\varepsilon}/3)\sigma_2(S^*)} < 0.62 < 1.$$

This indicates $\varepsilon_b < \hat{\varepsilon}$. Therefore, $\Lambda_2(S^*)\cdot\frac{n}{\theta_2} \leq (1+\varepsilon_b)\sigma(S^*)$ cannot be assured with $1-\delta'$ probability. Due to a similar reason, $\Lambda_2(S^o)\cdot\frac{n}{\theta_2} \geq (1-\varepsilon_c)\sigma(S^o)$ is also not assured with $1-\delta'$ probability. Therefore, D-SSA-Fix cannot provide any instance-specific approximation guarantee for the seed set S^* constructed from a given number of RR sets.