Revisiting the Stop-and-Stare Algorithms for Influence Maximization

Keke Huang[†], Sibo Wang[†], Glenn Bevilacqua[‡], Xiaokui Xiao[†], Laks V.S. Lakshmanan[‡]
[†]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[‡]Department of Computer Science, University of British Columbia, Canada
[†]{khuang005, swang, xkxiao}@ntu.edu.sg, [‡]{lennson,laks}@cs.ubc.ca

ABSTRACT

Influence maximization is a combinatorial optimization problem that finds important applications in viral marketing, feed recommendation, etc. Recent research has led to a number of scalable approximation algorithms for influence maximization, such as *TIM*$^+$ and *IMM*, and more recently, *SSA* and *D-SSA*. The goal of this paper is to conduct a rigorous theoretical and experimental analysis of *SSA* and *D-SSA* and compare them against the preceding algorithms. In doing so, we uncover inaccuracies in previously reported technical results on the accuracy and efficiency of *SSA* and *D-SSA*, which we set right. We also attempt to reproduce the original experiments on *SSA* and *D-SSA*, based on which we provide interesting empirical insights. Our evaluation confirms some results reported from the original experiments, but it also reveals anomalies in some other results and sheds light on the behavior of *SSA* and *D-SSA* in some important settings not considered previously. We also report on the performance of *SSA-Fix*, our modification to *SSA* in order to restore the approximation guarantee that was claimed for but not enjoyed by *SSA*. Overall, our study suggests that there exist opportunities for further scaling up influence maximization with approximation guarantees.

1. INTRODUCTION

Given a social network G, a *propagation model* M on how nodes in G may *influence* each other, and a positive integer k, the *influence maximization* problem asks for k nodes in G that can influence the largest number of nodes (in expectation) under the propagation model M. This problem is motivated by *viral marketing* [11, 24], in which a company provides free product samples to a number of influential individuals in a social network, aiming to advertise the products through these users via word of mouth. In addition, the problem has applications in feed recommendations and studies of propagations of rumors and innovations, etc. Solving influence maximization efficiently is an interesting and challenging problem that has attracted considerable interest [4–9, 12–14, 16–21, 25–27, 30] in the past decade. Most existing techniques, however, suffer from a tension between accuracy and efficiency. Specifically, the methods that provide non-trivial accuracy guarantees often require days to process even a small network, while the approaches that are practically efficient rely on heuristics and do not offer any assurance on the quality of the solutions returned.

The aforementioned tension is addressed in a recent line of research [5, 21, 26, 27] based on a subgraph sampling method developed by Borgs et al. [5]. (We elaborate this method in Section 2.2.) In particular, Borgs et al. [5] introduce the method in a theoretical study of influence maximization, and they utilize it to develop an algorithm that runs in $O(k(m+n)\epsilon^{-3}\log^2 n)$ expected time and returns an $(1-1/e-\epsilon)$-approximate solution for influence maximization with at least $1-1/n$ probability. Subsequently, Tang et al. [27] show that Borg et al.'s algorithm is inefficient in practice, and they propose a much more scalable algorithm (referred to as *TIM*$^+$) that not only provides the same approximation guarantee but also reduces the expected time complexity to $O(k(m+n)\epsilon^{-2}\log n)$. After that, Tang et al. [26] devise a further enhanced method (referred to as *IMM*) that retains *TIM*'s asymptotic guarantees on accuracy and efficiency but significantly decreases its computation time in practice. Most recently, Nguyen et al. [21] develop two algorithms, *SSA* and *D-SSA*, that improve over *IMM* in terms of empirical efficiency. They claim that *SSA* and *D-SSA* offer identical accuracy guarantees as *IMM* does, and that they are *asymptotically optimal* in terms of the numbers of subgraph samples that they use.

Unfortunately, we notice that the theoretical and experimental results reported by the proponents of *SSA* and *D-SSA* contain important gaps. Motivated by this, we conduct a rigorous theoretical and experimental analysis of *SSA* and *D-SSA* and set the story straight with respect to the state of the art for influence maximization. In doing so, we make the following contributions.

- **Refuting theoretical claims.** We revisit the key lemmas and theorems in [21] and identify substantial gaps in their proofs. Our analysis invalidates Nguyen et al.'s claims that (i) *SSA* and *D-SSA* return an $(1 - 1/e - \epsilon)$-approximation with at least $1 - 1/n$ probability and (ii) the numbers of subgraph samples generated by *SSA* and *D-SSA* are optimal.
- **Fixing *SSA* for accuracy.** We present a revised version of *SSA*, dubbed *SSA-Fix*, that restores $(1-1/e-\epsilon)$-approximation at the cost of increased computation overheads. We also implement and evaluate *SSA-Fix*.
- **Repeating experiments.** We successfully repeat some of the experiments in [21]. Our results support that the finding in [21] that *SSA* and *D-SSA* are more efficient than *IMM* when k is large.
- **Analyzing discrepancies.** We observe considerable discrepancies between some of our results and those in [21]. For example, our experiments show that *IMM* is consistently more efficient than *SSA* and *D-SSA* under the *independent cascade* model [17] when k is small, whereas the results in [21] demonstrate the op-

Proceedings of the VLDB Endowment, Vol. 10, No. 9
Copyright 2017 VLDB Endowment 2150-8097/17/05.

Table 1: Frequently used notations.

Notation	Description
$G(V, E)$	a social network G with node set V and edge set E
n, m	$n = \|V\|$ and $m = \|E\|$
$p(e)$	the propagation probability of an edge $e \in E$
$I(S)$	the spread of a seed set S in a propagation process
$\mathbb{E}[\cdot]$	the expected value of a random variable
OPT	the maximum expected spread of any size-k seed set
S_k	a size-k seed set
$\mathcal{R}$	a set of random RR sets
$Cov_{\mathcal{R}}(S_k)$	the number of RR sets in $\mathcal{R}$ that overlap S_k
$I_{est}(S_k)$	an estimation of S_k's expected spread output by Algorithm 3

posite. We analyze those discrepancies and conclude that the results in [21] are likely to be anomalous.

- **Providing new empirical insights.** We conduct new experiments with datasets and settings frequently used in the influence maximization literature but not considered in [21]. Based on the results, we shed new light on the behavior of *SSA* and *D-SSA*.

Overall, our study shows that *SSA* and *D-SSA* generally outperforms *IMM* in terms of empirical running time, but they fail to provide the worst-case guarantees claimed in [21]. This suggests that there exist opportunities to improve the state of the art for influence maximization by developing an algorithm that not only achieves the practical efficiency of *SSA* and *D-SSA* but also offer rigorous asymptotic guarantees in terms of both efficiency and accuracy.

Remark. We note that there are two revised versions of [21] submitted to arXiv on Sep. 7, 2016 [22] and Feb. 22, 2017 [23], respectively. The first version [22] was concurrent to our work, and it rectifies the accuracy issue of *SSA* in a way similar to our fix in Section 3.3. However, it does not discuss why the original *SSA* fails to provide the claimed accuracy guarantee, neither does it mention the misclaim of *SSA*'s efficiency which we analyze in Section 3.2. Furthermore, it does not point out the problems in *D-SSA*'s efficiency and accuracy claims which we elaborate in Section 4. The second version [23] was submitted after our work was made available to Nguyen et al., and it claims to have corrected the errors in the proofs for *SSA* and *D-SSA*. Unfortunately, we find that there still exist important gaps in some proofs in [23]. We refer interested readers to our technical report [1] for details.

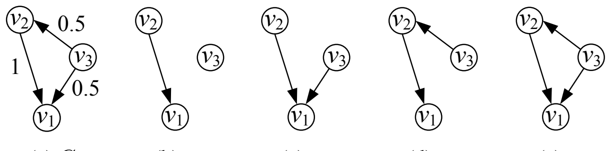
2. PRELIMINARIES

This section presents a formal definition of the influence maximization problem and summarizes the state of the art. For ease of exposition, we focus on the *independent cascade* model [17], which is one of the most well-adopted propagation models in the literature; nonetheless, our discussions can be easily extended to other propagation models, e.g., the *linear threshold* model [17]. Table 1 shows the notations that are frequently used.

2.1 Problem Definition

Let G be a social network with a node set V and a directed edge set E, with $|V| = n$ and $|E| = m$. For any two nodes u and v in G, if $\langle u, v\rangle \in E$, then we say that v is an outgoing neighbor of u and u is an incoming neighbor of v. Assume that each edge $e \in E$ is associated with a *propagation probability* $p(e) \in [0, 1]$.

Given a set S of nodes in G, an influence propagation process under the *independent cascade (IC)* model [17] is a discrete-time stochastic process as follows:



(a) G (b) g_1 (c) g_2 (d) g_3 (e) g_4

Figure 1: Input graph G and its corresponding random graphs.

1. At timestamp 0, all nodes in S are *activated*, while all other nodes remain *inactive*.
2. If a node u is first activated at timestamp i, then it has a chance to activate its outgoing neighbors at timestamp $i+1$, after which it can no longer activate any other node. In particular, if u has an inactive outgoing neighbor v, then u's activation attempt on v succeeds with probability $p(e)$ at timestamp $i + 1$, where $e = \langle u, v\rangle$ is the edge connecting u to v.
3. The influence propagation process terminates when none of the remaining inactivate nodes can be activated.

Let $I(S)$ be the number of activated nodes in G after the above process terminates. We refer to S as a *seed set* and $I(S)$ as its *spread* in the propagation process.

Given G and an integer $k > 0$, the influence maximization problem asks for a seed set S_k with the largest expected spread, i.e.,

$$S_k = \arg\max_{S':|S'|=k} \mathbb{E}[I(S')].$$

2.2 State of the Art

As mentioned in Section 1, the state-of-the-art methods for influence maximization [5,21,26,27] all utilize a sampling technique proposed by Borgs et al. [5]. This technique is based on the concept of *random reverse reachable (RR) set* [27], which we explain in the following: Let g be a directed graph obtained by removing each edge e in G with probability $1 - p(e)$ independently, and $\mathcal{G}$ be the distribution of g induced by the randomness in edge removal. Given an instance of g and a node v, the *reverse reachable (RR) 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 selected uniformly at random from G. We illustrate the above concepts with an example.

EXAMPLE 1. Figure 1a shows a social network with three nodes and three edges. (The number besides each edge e indicates its propagation probability $p(e)$.) If we independently remove each edge e in G with $1 - p(e)$ probability, then the resulting graph has four possibilities, which we illustrate as g_1, g_2, g_3, g_4 in Figures 1b-1e. The RR set for node v_1 in g_1 is $\{v_1, v_2\}$, since v_1 and v_2 are the only nodes that can reach v_1 in g_1. Meanwhile, the RR set for v_1 in any of g_2, g_3, and g_4 is $\{v_1, v_2, v_3\}$. □

Observe that if a node u appears in an RR set for another node v, then G must contain a directed path from u to v, which indicates that u has a probability to influence v in a propagation process. Based on this observation, Borgs et al. prove that random RR sets can be used to estimate the expected spread of any seed set, as shown in the following lemma:

LEMMA 1. *([5]) Let S be a seed set, g be a sample from G, and R be the RR set for a node v in g. Let ρ_1 be the probability that S can activate v in an influence propagation process, and ρ_2 be the probability that R overlaps with S. Then, $\rho_1 = \rho_2$.*

Let $\mathcal{R}$ be a set of random RR sets, and $Cov_{\mathcal{R}}(S)$ be the number of RR sets in $\mathcal{R}$ that overlap with S. By Lemma 1, $\frac{n}{|\mathcal{R}|}Cov_{\mathcal{R}}(S)$ can be used as an unbiased estimator of the expected spread of S:

COROLLARY 1. *([5])* $\mathbb{E}\left[\frac{n}{|\mathcal{R}|}Cov_{\mathcal{R}}(S)\right] = \mathbb{E}[I(S)]$.

Based on the above property of random RR sets, Borgs et al. [5] propose an elegant two-step method for influence maximization:

1. Generate a sufficiently large set $\mathcal{R}$ of random RR sets.
2. Use the greedy algorithm for maximum coverage [28] to identify a size-k seed set S_k that overlaps with a large number of RR sets in $\mathcal{R}$.

The intuition of the above method is that, if a seed set S intersects a large number of RR set in $\mathcal{R}$, then, by Corollary 1, the expected spread of S is likely to be large, in which case S could be a good solution for the influence maximization problem. Nevertheless, if we are to ensure the worst-case quality of S, then we have to carefully decide the size of $\mathcal{R}$. Towards this end, Borgs et al. propose to keep generating random RR sets in $\mathcal{R}$, until the total number of edges inspected during the generation process exceeds a predefined threshold τ. They show that when τ is properly set, there is at least $\frac{3}{5}$ probability that the method returns a seed set S with a $(1-1/e-\epsilon)$-approximation ratio, i.e.,

$$\mathbb{E}[I(S)] \geq (1-1/e-\epsilon)OPT,$$

where OPT denotes the maximum expected spread of any size-k seed set in G. In addition, Borgs et al. show that the method's success probability can be increased to $1-1/n$ at the cost of degrading its expected time complexity to $O(k(m+n)\epsilon^{-3}\log^2 n)$.

Following Borgs et al.'s work, Tang et al. [27] propose an improved solution, *TIM*$^+$, which runs in $O(k(m+n)\epsilon^{-2}\log n)$ expected time and returns a $(1-1/e-\epsilon)$-approximation with at least $1-1/n$ probability. Their idea is to avoid imposing a threshold on the number of edges inspected during the construction of $\mathcal{R}$, but to apply the Chernoff bounds to more accurately decide whether the number of random RR sets in $\mathcal{R}$ is sufficiently large. The experiments in [27] show that *TIM*$^+$ is several order of magnitude faster than Borgs et al.'s algorithm, and it is able to process a billion-edge social network in reasonable time. Subsequently, Tang et al. [26] present a further enhanced algorithm, *IMM*, which offers the same theoretical guarantee as *TIM*$^+$ does but provides much higher empirical efficiency. The improvement of *IMM* is mainly due to a martingale-based technique for deciding the size of $\mathcal{R}$ [26], which is shown to be less pessimistic than the method used in *TIM*$^+$.

Very recently, Nguyen et al. [21] observe that the computation time of *IMM* still leaves room for improvement, as it tends to generate an excessive number of RR sets when k is large. To address this issue, they propose two new algorithms, the *Stop-and-Stare algorithm (SSA)* and the *Dynamic Stop-and-Stare algorithm (D-SSA)*, both of which are similar in spirit to *IMM* but exploit the *zero-one estimation theorem* [10] to gauge the number of RR sets required. We will revisit *SSA* and *D-SSA* in Sections 3 and 4, respectively.

Besides the aforementioned work, there exists a plethora of other techniques [4, 6–9, 12, 14–20, 25, 29, 30] for influence maximization. The majority of these techniques [4,6–9,14,16,19,25,29,30] rely on heuristics to achieve practical efficiency, but fail to provide non-trivial worst-case guarantees in terms of the quality of results. For example, Wang et al. [29] presents a method that utilizes sketch-based heuristics to reduce the running time of *IMM*; however, as a trade off, the method neither retains the $(1-1/e-\epsilon)$-approximation ratio of *IMM* nor offers a non-trivial time complexity bound. Meanwhile, there are techniques [15, 17, 18, 20] that ensure $(1-1/e-\epsilon)$-approximations, but they incur significant overheads and do not scale to large graphs. In particular, among these methods, CELF++ [15] is shown to be the most efficient one under the IC model, and yet, its running time is several orders of magnitude larger than that of *IMM* [26].

Algorithm 1: SSA

```
   input : $G$, $\epsilon$, $\delta$ and $k$.
   output: A size-$k$ seed set $S_k$.
1  $\epsilon_1 = \frac{1}{6}\epsilon,\ \epsilon_2 = \frac{1}{2}\epsilon,\ \epsilon_3 = \frac{1}{4(1-1/e)}\epsilon$
2  $\Lambda_1 = \frac{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}{\epsilon_3^2}\ln(3/\delta)$
3  Generate a set $\mathcal{R}$ of $\Lambda_1$ random RR sets
4  repeat
5      $\langle S_k, Cov_{\mathcal{R}}(S_k)\rangle \leftarrow$ Max-Coverage($\mathcal{R}, k, n$)
6      if $Cov_{\mathcal{R}}(S_k) \geq \Lambda_1$ then
7          $I_{est}(S_k) \leftarrow$ Estimate-Inf$\left(G, S_k, \epsilon_2, \frac{\delta}{3}, \frac{1+\epsilon_2}{1-\epsilon_2}\cdot\frac{\epsilon_3^2}{\epsilon_2^2}\cdot|\mathcal{R}|\right)$
8          if $\frac{n}{|\mathcal{R}|}Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1)I_{est}(S_k)$ then
9              return $S_k$
10     Generate $|\mathcal{R}|$ random RR sets, and insert them into $R$
11 until $|\mathcal{R}| \geq \frac{8+2\epsilon}{\epsilon^2}\left(\ln\frac{2}{\delta} + \ln\binom{n}{k}\right)n$;
12 return $S_k$
```

3. SSA REVISITED

3.1 Description of SSA

In a nutshell, *SSA* runs in an iterative manner as follows:

1. Generate an initial set $\mathcal{R}$ of random RR sets.
2. Use the greedy approach for maximum coverage [28] to identify a size-k seed set S_k from $\mathcal{R}$.
3. Generate a separate set of random RR sets to test whether S_k is a good approximate solution.
4. If S_k is a good approximation, terminate the algorithm and return S_k; otherwise, double the number of random RR sets in $\mathcal{R}$, and goto Step 2.

The correctness and efficiency of *SSA* rely on Step 3, where it applies the *zero-one estimation theorem* [10] to decide whether a good approximation solution has been identified.

Algorithm 1 shows the details of *SSA*. It starts by initializing four internal parameters ϵ_1, ϵ_2, ϵ_3, and Λ_1 (Lines 1-2). After that, it generates an initial set $\mathcal{R}$ of random RR sets, with $|\mathcal{R}| = \Lambda_1$ (Line 3). The subsequent part of *SSA* consists of a number of iterations (Lines 4-11). In each iteration, *SSA* first invokes the standard greedy algorithm for maximum coverage (i.e., Algorithm 2) to obtain a size-k seed set S_k, as well as the number $Cov_{\mathcal{R}}(S_k)$ of RR sets in $\mathcal{R}$ that overlap S_k (Line 5). If $Cov_{\mathcal{R}}(S_k) \geq \Lambda_1$, then *SSA* invokes Algorithm 3 to derive an estimation $I_{est}(S)$ of the expected spread of S_k (Line 7). If $I_{est}(S)$ is sufficiently large, then *SSA* regards S_k as a good solution and returns it (Line 8-9); otherwise, *SSA* doubles the number of RR sets in $\mathcal{R}$, and proceeds to the next iteration (Line 10).

Nguyen et al. [21] claim that *SSA* returns a $(1-1/e-\epsilon)$-approximate solution with at least $1-\delta$ probability, but do not state its time complexity. Instead, they (i) consider a class of algorithms that utilize random RR sets for influence maximization, and (ii) claim that the number of random RR sets generated by *SSA* is asymptotically *optimal* within all algorithms in the class. As we analyze in Sections 3.2 and 3.3, however, the above claims of *SSA*'s accuracy and efficiency are both incorrect.

3.2 Misclaim on Efficiency

Consider any algorithm for influence maximization that derives its output S_k by applying the greedy approach for maximum coverage [28] on a set $\mathcal{R}$ of random RR sets. Let $\mathcal{C}$ be a subset of such

Algorithm 2: Max-Coverage

```
input : a set $\mathcal{R}$ of random RR sets, and $k$.
output: a size-$k$ seed set $S_k$, and the number $Cov_{\mathcal{R}}(S)$ of RR sets in
        $\mathcal{R}$ that overlap $S_k$.
1 $S_k = \emptyset$
2 for $i = 1$ to $k$ do
3     $v = \arg\max_{v' \in V} (Cov_{\mathcal{R}}(S_k \cup \{v'\}) - Cov_{\mathcal{R}}(S_k))$
4     Insert $v$ into $S_k$
5 return $\langle S_k, Cov_{\mathcal{R}}(S_k) \rangle$
```

Algorithm 3: Estimate-Inf

```
input : $G$, $\epsilon_2$, $\delta_2$, a seed set $S$, and a threshold $T_{max}$.
output: an estimation $I_{est}(S)$ of the expected spread of $S$.
1 $\Lambda_2 = 1 + \frac{(4e-8)(1+\epsilon_2)}{\epsilon_2^2} \ln \frac{1}{\delta_2}$
2 $cnt = 0$
3 for $i = 1$ to $T_{max}$ do
4     Generate a random RR set $R$
5     if $R \cap S \neq \emptyset$ then
6         $cnt = cnt + 1$
7     if $cnt \geq \Lambda_2$ then
8         return $n \cdot cnt / i$
9 return $-1$
```

algorithms that ensure the following two inequalities:

$$Pr\left[\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_a)\mathbb{E}[I(S_k)]\right] \geq 1 - \delta_a, \quad (1)$$

$$Pr\left[\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k^\circ) \geq (1-\epsilon_b) OPT\right] \geq 1 - \delta_b, \quad (2)$$

where S_k° denotes the optimal solution for the influence maximization problem, and ϵ_a, ϵ_b, δ_a, and δ_b are constants. Let $N_{min}^1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ be the minimum size of $\mathcal{R}$ needed to ensure both Equations 1 and 2. Nguyen et al. [21] make the following claim regarding *SSA*'s efficiency (see Lemma 9 in [21]):

CLAIM 1 ([21]). *With at least $(1-\delta)$-probability, the number of random RR sets generated by SSA is at most a constant times $N_{min}^1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$, with $\epsilon_a = \frac{2}{3}\epsilon + \frac{1}{12}\epsilon^2$, $\epsilon_b = \frac{1}{4(1-1/e)}\epsilon$, $\delta_a = \delta/3$, and $\delta_b = 2\delta/3$.*

To prove Claim 1, Nguyen et al. [21] point out that *SSA* terminates when the conditions in Lines 6 and 8 in Algorithm 1 simultaneously hold, i.e.,

$$Cov_{\mathcal{R}}(S_k) \geq \Lambda_1, \text{and} \quad (3)$$

$$\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1) I_{est}(S_k), \quad (4)$$

where $\Lambda_1 = (4e-8)(1+\epsilon_1)(1+\epsilon_2)\ln(3/\delta)/\epsilon_3^2$, and S_k is obtained by applying the greedy maximum coverage approach on $\mathcal{R}$. Nguyen et al. [21] claim that, after the size $\mathcal{R}$ reaches a constant times $N_{min}^1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$, there is a high probability that both Equations 3 and 4 are satisfied, i.e., Claim 1 holds.

In particular, Nguyen et al. make the following claim in connection to Equation 3 (see Lemma 7 in [21]):

CLAIM 2 ([21]). *There exists a constant c_1, such that when $|\mathcal{R}| \geq \frac{c_1}{1-1/e-\epsilon} N_{min}^1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$, Equation 3 holds with at least $1-\delta_b$ probability.*

In addition, they claim the following regarding Equation 4 (see Lemma 8 in [21]):

In this case, the necessary number of RR sets is the same as that to satisfy Equation 1 when $\epsilon_a = \epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2$. The only question here is whether the *Estimate-Inf* algorithm succeeds and returns a good estimation of $\mathbb{E}[I(S_k)]$ when the main algorithm has $|\mathcal{R}|$ random RR sets.

...

(Note: The subsequent part of the proof shows that Estimate-Inf can return a good estimation of $\mathbb{E}[I(S_k)]$*.)*

Figure 2: Nguyen et al.'s proof for Claim 3 in [21].

CLAIM 3 ([21]). *When $|\mathcal{R}| \geq \frac{c_1}{1-1/e-\epsilon} N_{min}^1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$, Equation 4 holds with at least $1-\delta_a$ probability.*

However, the proof of Claim 3 is incorrect, as we discuss in the following. Figure 2 shows an abridged version of Nguyen et al.'s proof for Claim 3. The proof begins by claiming that, if the number of random RR sets is large enough to ensure Equation 1, then it is also sufficient to ensure Equation 4. However, this is true only if Equation 1 implies Equation 4, i.e., when

$$\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_a)\mathbb{E}[I(S_k)] \quad (5)$$

$$\implies \frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1) I_{est}(S_k). \quad (6)$$

In turn, this requires the right hand side of Equation 5 to be no larger than the right hand side of Equation 6, i.e.,

$$(1+\epsilon_a)\mathbb{E}[I(S_k)] \leq (1+\epsilon_1) I_{est}(S_k),$$

which is equivalent to

$$(1+\epsilon_2)\mathbb{E}[I(S_k)] \leq I_{est}(S_k), \quad (7)$$

since $(1+\epsilon_a) = (1+\epsilon_1)(1+\epsilon_2)$.

Recall that $I_{est}(S_k)$ is the output of *Estimate-Inf*. By Nguyen et al.'s proof in [21], *Estimate-Inf* ensures that $I_{est}(S_k) \leq (1+\epsilon_2)\mathbb{E}[I(S_k)]$ holds with at least $1-\delta_a$ probability. That is, Equation 7 holds with at most δ_a probability, and hence, Equation 5 does not imply Equation 6. Consequently, Claim 3 does not hold, which in turn invalidates the efficiency guarantee of *SSA* in Claim 1.

3.3 Misclaim on Accuracy and A Fix

To establish the approximation guarantee of *SSA*, Nguyen et al. [21] leverage the fact that Algorithm 1 terminates when both Equations 3 and 4 (which correspond to Lines 6 and 8 in Algorithm 1, respectively) are satisfied, and they analyze the quality of the size-k seed set S_k based on Equations 3 and 4. In particular, they first consider the estimation $I_{est}(S_k)$ of S_k's expected spread obtained in Line 7 of Algorithm 1, and show the following corollary (see Corollary 1 in [21]):

COROLLARY 2. *Let $I_{est}(S_k)$ be the output of Estimate-Inf in Line 7 of Algorithm 1. Then,*

$$\Pr\left[I_{est}(S_k) \leq (1+\epsilon_2)\mathbb{E}[I(S_k)]\right] \geq 1 - \frac{\delta}{3}.$$

Based on Corollary 2 and Line 8 of Algorithm 1, Nguyen et al. make the following claim (see Lemma 5 in [21]):

CLAIM 4. *SSA returns a seed set S_k with*

$$\Pr\left[\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1)(1+\epsilon_2)\mathbb{E}[I(S_k)]\right] \geq 1 - \frac{\delta}{3}. \quad (8)$$

Then, based on Claim 4, they proceed to prove that *SSA* returns a $(1-1/e-\epsilon)$-approximation with at least $1-\delta$.

Recall that *SSA* terminates when the selected seed set S_k satisfies two conditions:

$$Cov_{\mathcal{R}}(S_k) \geq \Lambda_1, \text{ and}$$
$$\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1) I_{est}(S_k).$$

From Corollary 2, we have

$$\Pr\left[I_{est}(S_k) \leq (1+\epsilon_2)\mathbb{E}[I(S_k)]\right] \geq 1-\delta/3.$$

By this and $\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1) I_{est}(S_k)$, we have

$$\Pr\left[\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1)(1+\epsilon_2)\mathbb{E}[I(S_k)]\right] \geq 1-\delta/3.$$

This completes the proof.

Figure 3: Nguyen et al.'s proof for Claim 4 [21].

Unfortunately, Claim 4 is incorrect. To help establish this, we show in Figure 3 Nguyen et al.'s proof of Claim 4. In short, Nguyen et al. prove that Equation 8 holds when $\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1) I_{est}(S_k)$ (see Line 8 in Algorithm 1) and

$$I_{est}(S_k) \leq (1+\epsilon_2)\mathbb{E}[I(S_k)]. \quad (9)$$

Based on this, they assert that Claim 4 is valid because, by Corollary 2, Equation 9 holds with at least $1-\delta/3$ probability. This assertion, however, overlooks the fact that *SSA* consists of several iterations, and that each iteration generates a different version of $I_{est}(S_k)$ (see Line 7 in Algorithm 1). If a certain iteration produces a version of $I_{est}(S_k)$ that violates Equation 9, then *SSA* may terminate in that iteration and return a size-k seed set S_k that violates Equation 8. To avoid this issue, it is necessary that Equation 9 holds in all iterations of *SSA*. In contrast, the proof in Figure 3 only considers the case when Equation 9 holds in the last iteration of *SSA*, but ignores all previous iterations. As a consequence, the proof does not establish Claim 4, which invalidates Nguyen et al.'s claim on the approximation guarantee of *SSA*.

One way to address the above problem is to ensure that Equation 9 holds with at least $1-\delta/3/x$ probability in each iteration of *SSA*, where x denotes the total number of iterations. Then, by the union bound, there is at least $1-\frac{\delta}{3}$ probability that Equation 9 is valid for all iterations of *SSA*, in which case Claim 4 holds. For this purpose, we can change Line 7 in Algorithm 1 to

$$I_{est}(S_k) \leftarrow \text{ Estimate-Inf}\left(G, S_k, \epsilon_2, \frac{\delta}{3\alpha}, \frac{(1+\epsilon_2)\epsilon_3^2}{(1-\epsilon_2)\epsilon_2^2}|\mathcal{R}|\right), \quad (10)$$

where α is an upper bound of the number of iterations in *SSA*. To set α to an appropriate value, we observe that (i) *SSA* starts with a set $\mathcal{R}$ of Λ_1 RR sets in its first iteration (where Λ_1 is as defined in Line 2 of Algorithm 1) and doubles the size of $\mathcal{R}$ in each subsequent iteration, and (ii) *SSA* terminates if the size of $\mathcal{R}$ exceeds $\frac{8+2\epsilon}{\epsilon^2}\left(\ln\frac{2}{\delta}+\ln\binom{n}{k}\right)n$ (see Line 11 of Algorithm 1). Accordingly, we set $\alpha = \log_2\left(\frac{8+2\epsilon}{\epsilon^2}\left(\ln\frac{2}{\delta}+\ln\binom{n}{k}\right)\frac{n}{\Lambda_1}\right)$, which ensures that α is no smaller than the number of iterations in *SSA*. Interested readers are referred to our technical report [1] for a detailed proof on the correctness of this fix.

4. D-SSA REVISITED

4.1 Description of D-SSA

SSA (i.e., Algorithm 1) has three internal parameters ϵ_1, ϵ_2, and ϵ_3. These three parameters (i) decide the approximation errors allowed in its estimation steps (see Lines 7 and 8 in Algorithm 1), and (ii) determine the number of random RR sets generated in each iteration of *SSA*. In *SSA*, they are fixed to $\epsilon_1 = \frac{1}{6}\epsilon$, $\epsilon_2 = \frac{1}{2}\epsilon$, $\epsilon_3 = \frac{1}{4(1-1/e)}\epsilon$. Nguyen et al. [21] suggest that fixing them in all iterations may lead to suboptimal performance, and propose to vary them in different iterations to reduce the total number of random RR sets generated. Accordingly, they propose the *D-SSA* method, as shown in Algorithm 4.

Algorithm 4: D-SSA

```
   input : $G$, $\epsilon$, $\delta$ and $k$.
   output: A size-$k$ seed set $S_k$.
1  $\Lambda = \frac{(4e-8)(1+\epsilon)^2}{\epsilon^2}\ln\frac{2}{\delta}$
2  Generate a set $\mathcal{R}$ of $\Lambda$ random RR sets
3  repeat
4      $\langle S_k, Cov_{\mathcal{R}}(S_k)\rangle \leftarrow$ Max-Coverage($\mathcal{R}, k, n$)
5      Generate a set $\mathcal{R}'$ of $|\mathcal{R}|$ random RR sets
6      $I_{est}(S_k) \leftarrow \frac{n}{|\mathcal{R}'|}\cdot Cov_{\mathcal{R}'}(S_k)$
7      $\epsilon_1 = \frac{n}{|\mathcal{R}|}\frac{Cov_{\mathcal{R}}(S_k)}{I_{est}(S_k)} - 1$
8      if $\epsilon_1 \leq \epsilon$ then
9          $\epsilon_2 = \frac{\epsilon-\epsilon_1}{2(1+\epsilon_1)}$, $\epsilon_3 = \frac{\epsilon-\epsilon_1}{2(1-1/e)}$
10         $\delta_1 = e^{-\frac{\epsilon_3^2\cdot Cov_{\mathcal{R}}(S_k)}{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}}$
11         $\delta_2 = e^{-\frac{\epsilon_2^2\cdot(Cov_{\mathcal{R}'}(S_k)-1)}{(4e-8)(1+\epsilon_2)}}$
12         if $\delta_1+\delta_2 \leq \delta$ then
13             return $S_k$
14     $\mathcal{R} = \mathcal{R}\cup\mathcal{R}'$
15 until $|\mathcal{R}| \geq \frac{8+2\epsilon}{\epsilon^2}\left(\ln\frac{2}{\delta}+\ln\binom{n}{k}\right)n$;
16 return $S_k$
```

Similar to *SSA*, *D-SSA* also generates an initial set $\mathcal{R}$ of random RR sets (Line 1 in Algorithm 4, and then enters an iterative process (Lines 3-15). In each iteration, *D-SSA* first applies the greedy approach for maximum coverage on $\mathcal{R}$ to obtain a size-k seed set S_k, along with the number $Cov_{\mathcal{R}}(S_k)$ of RR sets in $\mathcal{R}$ that overlap S_k (Line 4). After that, it generates another set $\mathcal{R}'$ of random RR sets with $|\mathcal{R}'| = |\mathcal{R}|$ (Line 5). Then, it uses $\mathcal{R}'$ to derive an estimation $I_{est}(S_k)$ of S_k's expected spread, based on which it determines the value of ϵ_1 (Lines 5-7). If $\epsilon_1 \leq \epsilon$, then the algorithm proceeds to decide the values of ϵ_2 and ϵ_3 based on ϵ_1 and ϵ (Line 9). In addition, it also derives δ_1 and δ_2, which quantify the probabilities that $\mathcal{R}$ and $\mathcal{R}'$, respectively, do not provide the accurate estimations required by *D-SSA* (Lines 10-11). If $\delta_1+\delta_2 \leq \delta$, then *D-SSA* returns S_k as a solution (Lines 12-13); otherwise, it doubles the size of $\mathcal{R}$ by moving all RR sets in $\mathcal{R}'$ into $\mathcal{R}$, after which it proceeds to the next iteration (Line 14).

Nguyen et al. [21] claim that *D-SSA* achieves $(1-1/e-\epsilon)$-approximation with at least $1-\delta$ probability. In addition, they claim that the number of random RR sets generated by *D-SSA* is asymptotically *optimal* among a large class of algorithms that utilize random RR sets. However, both claims are incorrect, as we show in Sections 4.2 and 4.3.

4.2 Misclaim on Efficiency

Let $N^1_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ be as defined in Section 3, i.e., $N^1_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ is the minimum size of a set $\mathcal{R}$ of random RR sets that simultaneously ensure Equations 1 and 2. Let $N^2_{min}(\epsilon, \delta)$ be the minimum value of $N^1_{min}(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ under the constraints that $\epsilon_a + (1-1/e)\epsilon_b \leq \epsilon$ and $\delta_1 + \delta_2 \leq \delta$. Nguyen et al. [21] point out that $N^2_{min}(\epsilon, \delta)$ is a lower bound on the expected number of random RR sets required by the state-of-the-art solutions [5,26,27]. In addition, they make the following claim on the efficiency of *D-SSA* (see Theorem 5 in [21]):

(Note: In the following, Equations 11, 12-14, and 15 correspond to Equations 64, 75-78 in [21], respectively.)

$$\cdots$$

$$\Pr\left[|\mathcal{R}'| \geq \frac{\Upsilon(\frac{\epsilon_b^*}{2}, \frac{\delta_b^*}{2})}{E[I(S)]/n}\right] \geq 1 - \delta_a^* - \frac{\delta_b^*}{2}. \tag{11}$$

$$\cdots$$

$$Cov_{\mathcal{R}}(S_k) \geq (4e-8)\frac{1}{\epsilon_b^{*2}} \ln \frac{2}{\delta_b^*}. \tag{12}$$

Combining Equation 12 and the definition of δ_1 in *D-SSA*, we get

$$\delta_1 = e^{-\frac{Cov_{\mathcal{R}}(S_k)\epsilon_3^2}{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}} \leq e^{-\frac{(4e-8)\ln\frac{2}{\delta_b^*}\frac{1}{\epsilon_3^2}\cdot\epsilon_3^2}{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}} \leq \delta_b^*/2. \tag{13}$$

Based on the fact $|\mathcal{R}'| = |\mathcal{R}|$ and Equation 11, we get

$$\delta_2 = e^{-\frac{Cov_{\mathcal{R}'}(S_k)\epsilon_2^2}{(4e-8)(1+\epsilon_2)}} \leq e^{-\frac{(4e-8)\ln\frac{2}{\delta_b^*}\frac{1}{\epsilon_2^2}\cdot\epsilon_2^2}{(4e-8)(1+\epsilon_2)}} \leq \delta_b^*/2. \tag{14}$$

From Equation 13 and Equation 14, we conclude

$$\delta_1 + \delta_2 \leq \delta_b^*/2 + \delta_b^*/2 = \delta_b^* \leq \delta, \tag{15}$$

which proves the claim.

Figure 4: Nguyen et al.'s proof for Claim 6 in [21].

CLAIM 5. *([21]) With at least $1-\delta$ probability, the number of random RR sets generated by D-SSA is at most a constant times $N_{min}^2(\epsilon, \delta)$.*

To establish Claim 5, Nguyen et al. [21] consider the conditions on which *D-SSA* terminates, and aim to quantify the number of random RR sets required by *D-SSA* to satisfy those conditions. As shown in Lines 12 in Algorithm 4, one of *D-SSA*'s termination conditions is $\delta_1 + \delta_2 \leq \delta$. In connection to this, Nguyen et al. [21] make the following claim (see Lemma 12 in [21]):

CLAIM 6 ([21]). *Suppose that when $\epsilon_a = \epsilon_a^*$, $\epsilon_b = \epsilon_b^*$, $\delta_a = \delta_a^*$, and $\delta_b = \delta_b^*$, we have $N_{min}^2(\epsilon, \delta) = N_{min}^1(\epsilon_a, \epsilon_b, \delta_a, \delta_b)$ with $\epsilon_a + (1-1/e)\epsilon_b \leq \epsilon$ and $\delta_1 + \delta_2 \leq \delta$. In addition, define M as*

$$M = \frac{\ln\frac{2}{\delta_b^*}}{\ln\frac{1}{\delta_b^*}} \frac{4(1+\epsilon_a^*)^2}{(\frac{1}{2} - \frac{1}{e} - \frac{\epsilon}{2})^3}.$$

Then, there exists a constant c_1 such that if $|\mathcal{R}| \geq Mc_1 N_{min}^2(\epsilon, \delta)$ and $\epsilon_1 \leq \frac{\epsilon_a^* + \epsilon_b^*/2}{1 - \epsilon_b^*/2}$, then we have $\delta_1 + \delta_2 \leq \delta$.

Unfortunately, Claim 6 is invalid due to two gaps in its proof; in turn, this nullifies the efficiency guarantee of *D-SSA* stated in Claim 5. To clarify this, we show in Figure 4 a part of Nguyen et al.'s proof for Claim 6, and we will illustrate that Equations 13 and 14 are incorrect. In particular, based on the first inequality in Equation 13,

$$\delta_1 \leq e^{-\frac{(4e-8)\ln\frac{2}{\delta_b^*}\frac{1}{\epsilon_3^2}\cdot\epsilon_3^2}{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}} = e^{-\frac{\ln\frac{2}{\delta_b^*}}{(1+\epsilon_1)(1+\epsilon_2)}}.$$

Then, by the fact that $(1+\epsilon_1)(1+\epsilon_2) \geq 1$, we have

$$e^{-\frac{\ln\frac{2}{\delta_b^*}}{(1+\epsilon_1)(1+\epsilon_2)}} \geq e^{-\ln\frac{2}{\delta_b^*}} = \delta_b^*/2.$$

This contradicts the second inequality in Equation 13, i.e.,

$$e^{-\frac{(4e-8)\ln\frac{2}{\delta_b^*}\frac{1}{\epsilon_3^2}\cdot\epsilon_3^2}{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}} \leq \delta_b^*/2.$$

Similarly, based on the first inequality of Equation 14 and the fact that $1+\epsilon_2 \geq 1$, we have

$$\delta_2 \leq e^{-\frac{(4e-8)\ln\frac{2}{\delta_b^*}\frac{1}{\epsilon_2^2}\cdot\epsilon_2^2}{(4e-8)(1+\epsilon_2)}} = e^{-\frac{\ln\frac{2}{\delta_b^*}}{1+\epsilon_2}} \geq e^{-\ln\frac{2}{\delta_b^*}} = \delta_b^*/2,$$

which contradicts the second inequality in Equation 14, i.e.,

$$e^{-\frac{(4e-8)\ln\frac{2}{\delta_b^*}\frac{1}{\epsilon_2^2}\cdot\epsilon_2^2}{(4e-8)(1+\epsilon_2)}} \leq \delta_b^*/2.$$

Since Equations 13 and 14 are invalid, Equation 15 does not hold.

4.3 Misclaim on Accuracy

Nguyen et al. [21] make the following claim about the approximation guarantee of *D-SSA* (see Theorem 4 in [21]):

CLAIM 7. *Given a graph G, $0 \leq \epsilon \leq 1 - 2/e$, and $0 \leq \delta \leq 1$ as inputs, D-SSA returns a $(1-1/e-\epsilon)$-approximate solution.*

Figure 5 shows Nguyen et al.'s proof for Claim 7. Its basic idea is to prove that when *D-SSA* terminates, the seed set that it returns is as least as good as that returned by *SSA*. However, the proof is incorrect in at least three aspects. First, as we shown in Section 3.3, *SSA* itself does not ensure $(1-1/e-\epsilon)$-approximation with at least $1-\delta$ probability; therefore, Claim 7 cannot be established by proving that *D-SSA* provides the same approximation as *SSA* does. Second, the proof claims that Equation 19 (see Figure 5) is equivalent to the second stopping condition in *SSA*, i.e., Equation 4 (see Line 8 in Algorithm 1). However, if we compare Equation 19 with Equation 4, it is apparent that Equation 19 contains a typo and should have been

$$\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1)(1+\epsilon_2)\mathbb{E}[I(S_k)]. \tag{16}$$

In what follows, we will use Equation 16 as a replacement of Equation 19 in our discussion.

The third (and most serious) bug in the proof lies in the derivation of Equation 16. As shown in Figure 5, the proof claims that Equation 16 follows from Equations 17 and 18, by drawing connection between Equation 16 and the *Estimate-Inf* procedure (i.e., Algorithm 3). This could be a bit confusing since *Estimate-Inf* is not invoked in *D-SSA*. To clarify this, we first elaborate Nguyen et al.'s reasoning in the derivation of Equation 16, and then we point out its loophole.

Nguyen et al.'s derivation of Equation 16. We first introduce the *zero-one estimator theorem* [10], which is used by Nguyen et al. [21] to establish the accuracy bound of *Estimate-Inf* in Corollary 2; for ease of exposition, we present a simplified version of the theorem in the context of our discussion.

THEOREM 1. *([10]) Let ϵ and δ be two constants in $(0,1)$, and S be any seed set. Suppose that we generate a sequence of random RR sets, until the number of RR sets that overlap with S is at least*

$$1 + \frac{(4e-8)(1+\epsilon)}{\epsilon^2} \ln \frac{1}{\delta}.$$

Let $\mathcal{R}^*$ be the set of RR sets thus obtained. Then,

$$\Pr\left[\frac{n}{|\mathcal{R}^*|} Cov_{\mathcal{R}^*}(S) \leq (1+\epsilon)\cdot\mathbb{E}[I(S)]\right] \geq 1-\delta. \tag{20}$$

The proof in Figure 5 implies that Equation 16 can be derived from Theorem 1 as follows. By Theorem 1, given constants ϵ_2 and

(Note: In the following, Equations 17, 18, and 19 correspond to Equations 26, 27, and 28 in [21], respectively.)

We prove that the two stopping conditions in *SSA* still hold in *D-SSA*, and thus, *D-SSA* has the same approximation factor as *SSA* does. Directly from Algorithm 4, when *D-SSA* terminates, the following conditions are satisfied:

$$Cov_{\mathcal{R}}(S_k) = \frac{(4e-8)(1+\epsilon_1)(1+\epsilon_2)}{\epsilon_3^2} \ln \frac{1}{\delta_1} \quad (17)$$

$$Cov_{\mathcal{R}'}(S_k) = 1 + \frac{(4e-8)(1+\epsilon_2)}{\epsilon_2^2} \ln \frac{1}{\delta_2} \quad (18)$$

with $\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2 + (1-1/e)\epsilon_3 = \epsilon$ and $\delta_1 + \delta_2 = \delta$. Equation 17 is the first stopping condition in *SSA*. Equation 18 is the checking condition in the *Estimate-Inf* procedure in *SSA*. Together with the setting of ϵ_1, we obtain the second stopping condition in *SSA*:

$$\frac{n}{|\mathcal{R}|} Cov_{\mathcal{R}}(S_k) \geq (1+\epsilon_1)(1+\epsilon_2)\mathbb{E}[I(S_k)]. \quad (19)$$

Thus, the $(1-1/e-\epsilon)$-approximation factor is followed from *SSA*, which completes the proof.

Figure 5: Nguyen et al.'s proof for Claim 7 in [21].

δ_2 in $(0,1)$ and a size-k seed set S_k, if we generate a set $\mathcal{R}'$ of random RR sets that satisfies Equation 18, then we have

$$\Pr\left[\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}'}(S_k) \leq (1+\epsilon_2) \cdot \mathbb{E}[I(S_k)]\right] \geq 1 - \delta_2. \quad (21)$$

Meanwhile, by Lines 5-7 in Algorithm 4, we have

$$\epsilon_1 = \frac{n}{|\mathcal{R}|} \frac{Cov_{\mathcal{R}}(S_k)}{I_{est}(S_k)} - 1 = \frac{Cov_{\mathcal{R}}(S_k)}{Cov_{\mathcal{R}'}(S_k)} - 1. \quad (22)$$

Combining Equations 21 and 22, we have

$$\Pr\left[\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}}(S_k) \leq (1+\epsilon_1)(1+\epsilon_2)\mathbb{E}[I(S_k)]\right] \geq 1 - \delta_2.$$

Namely, Equation 16 holds with at least $1-\delta_2$ probability.

Loophole in derivation. The above derivation, however, requires that ϵ_2 is a *constant* independent of $\mathcal{R}'$, due to the requirements in Theorem 1. Unfortunately, *D-SSA* sets

$$\epsilon_2 = \frac{\epsilon - \epsilon_1}{2(1+\epsilon_1)} = \frac{1+\epsilon}{2} \cdot \frac{Cov_{\mathcal{R}'}(S_k)}{Cov_{\mathcal{R}}(S_k)} - \frac{1}{2}, \quad (23)$$

which is a variable that depends on the set $\mathcal{R}'$ of random RR sets on which Theorem 1 is applied. This invalidates Equation 21, which in turn nullifies Claim 7.

To illustrate the problem of setting ϵ_2 based on Equation 23, one may consider the probabilistic event in Equation 21:

$$\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}'}(S_k) \leq (1+\epsilon_2) \cdot \mathbb{E}[I(S_k)], \quad (24)$$

namely, the event that the random variable $\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}'}(S_k)$ is at most $1+\epsilon_2$ times its expectation $\mathbb{E}[I(S_k)]$. When ϵ_2 is a constant, this event is of the same type considered in Theorem 1, and thus, its probability can be bounded using the latter. However, if ϵ_2 is as defined in Equation 23, then the above event becomes

$$\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}'}(S_k) \leq \left(\frac{1+\epsilon}{2} \cdot \frac{Cov_{\mathcal{R}'}(S_k)}{Cov_{\mathcal{R}}(S_k)} + \frac{1}{2}\right) \cdot \mathbb{E}[I(S_k)].$$

Note that the random variable $Cov_{\mathcal{R}'}(S_k)$ appears in both sides of the inequality. By moving $Cov_{\mathcal{R}'}(S_k)$ to the left hand side of inequality, we have

Table 2: Dataset details. ($\mathbf{K = 10^3, M = 10^6, G = 10^9}$)

Dataset	n	m	Type	Avg. degree
NetHEPT	15.2K	31.4K	undirected	4.18
NetPHY	37.2K	181K	undirected	9.73
Enron	36.7K	184K	undirected	10.0
Epinions	132K	841K	directed	13.4
DBLP	655K	1.99M	undirected	6.08
Orkut	3.07M	117M	undirected	76.2
LiveJournal	4.85M	69.0M	directed	28.5
Twitter	41.7M	1.47G	directed	70.5

$$\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}'}(S_k) \leq \left(\frac{1}{2 - \frac{|\mathcal{R}'|}{n} \frac{(1+\epsilon)\mathbb{E}[I(S_k)]}{Cov_{\mathcal{R}}(S_k)}}\right) \mathbb{E}[I(S_k)], \quad (25)$$

which represents the event that the random variable $\frac{n}{|\mathcal{R}'|} Cov_{\mathcal{R}'}(S_k)$ is at most $\left(2 - \frac{|\mathcal{R}'|}{n} \frac{(1+\epsilon)\mathbb{E}[I(S_k)]}{Cov_{\mathcal{R}}(S_k)}\right)^{-1}$ times its expectation $\mathbb{E}[I(S_k)]$. The probability of this event cannot be bounded using Theorem 1 whenever $\frac{|\mathcal{R}'|}{n} \frac{(1+\epsilon)\mathbb{E}[I(S_k)]}{Cov_{\mathcal{R}}(S_k)} < 1$.

We note that the above loophole can be fixed only if we set ϵ_2 to a constant in *D-SSA*. However, it is unclear how this can be done without changing *D-SSA* in a substantial manner.

5. EXPERIMENTS

This section repeats and extends the experimental evaluation of *SSA* and *D-SSA* in [21]. All of our experiments are conducted on a Linux machine with an Intel Xeon 2.6GHz CPU and 64GB RAM. In each experiment, we measure the performance of each method 20 times and report the average measurement.

5.1 Experimental Setting

Datasets. The experiments in [21] are based on eight datasets, i.e., NetHEPT, NetPHY, Enron, Epinions, DBLP, Orkut, Twitter, and Friendster. We use all of these datasets except for Friendster, since every tested method runs out of memory on Friendster on our machine. (This issue does not occur in the experiments in [21] since the machine used has 100GB RAM.) As an alternative, we replace Friendster with another dataset, LiveJournal, which is frequently used in the literature. Table 2 shows the details of our datasets.

Algorithms. We evaluate four algorithms: *SSA* [21], *D-SSA* [21], *IMM* [26], and *SSA-Fix*, which is a revised version of *SSA* that returns $(1-1/e-\epsilon)$-approximation with at least $1-\delta$ probability, as we explain in Section 3.3. We obtain the C++ implementation of *IMM* from [2] and those of *SSA* and *D-SSA* from [3]. The *SSA* and *D-SSA* implementations deviate from the pseudo-code [21] in several minor places (e.g., the constant Λ_1 in Algorithm 1); to repeat the experiments in [21], we modify the implementations of *SSA* and *D-SSA* to make them consistent with the descriptions in [21][1].

In addition, we observe that the *SSA* and *D-SSA* implementations are more optimized than that of *IMM* since (i) they adopt a faster random number generator and a more efficient method to produce random RR sets under the LT model (see Appendix A for a discussion), and (ii) they use a more optimized implementation of the greedy algorithm for maximum coverage [28] to identify seed sets from random RR sets. To eliminate the effects of these implementation differences from our comparison of *IMM*, *SSA*, and *D-SSA*, we

[1]We have tested the performance of the implementations before and after our consistency-driven modifications. In general, the *SSA* and *D-SSA* implementations run around 10% faster after our modifications for $k \geq 2500$, and around 15% slower for $k \leq 50$.

revise the *IMM* code so that it uses the random number generator as well as the methods for RR set generation and maximum coverage in the *SSA* and *D-SSA* code. We refer to the revised implementation of *IMM* as *IMM**.

Parameter settings. Following [21], we evaluate all algorithms with $\epsilon = 0.1$ and $\delta = 1/n$ under two popular influence propagation models: *independent cascade (IC)* [17] and *linear threshold (LT)* [17]. Note that this setting is to the advantage of *SSA* and *D-SSA* because, as we point out in Sections 3 and 4, they do not guarantee $(1 - 1/e - \epsilon)$-approximation with at least $1 - \delta$ probability, while *IMM* and *SSA-Fix* do. Following previous work [21], for each edge $e = (u, v)$, we set its propagation probability $p(e)$ to $\frac{1}{d_{in}(v)}$, where $d_{in}(v)$ is the in-degree of the node v.

We consider two settings of the size k of the seed set: (i) $k \in \{1, 2500, 5000, 7500, \ldots, 20000\}$, referred to as the *large k setting*, and (ii) $k \in \{1, 2, 5, 10, 20, 30, 40, 50\}$, referred to as the *small k setting*. The large k setting is used in [21], while the small k setting is adopted in most existing work on influence maximization. We note that the small k setting is more consistent with the motivation of influence maximization, i.e., a marketer aims to provide product samples to *a small number of individuals* to start a viral marketing campaign based on word-of-month; nonetheless, we recognize that there might be other applications where the large k setting could be useful.

5.2 Results under the IC model

In the first set of experiments, we examine the performance of *SSA*, *D-SSA*, *SSA-Fix*, and *IMM* under the IC model. Figures 6 and 8 report the running time of each method under the large and small k settings, respectively. In addition, Figure 7 shows the number of RR sets that each method generates. The experiments in Figures 6e-6h, 7, and 8 are new with respect to [21], since (i) [21] does not report any result under the IC model concerning the small k setting or the number of RR sets, and (ii) it omits Enron, Epinions, Orkut, and LiveJournal under the IC model. The maximum value of k tested on NetHEPT is 15000, since NetHEPT has only 15233 nodes. In what follows, we analyze our results based on (i) whether the same experiments are conducted in [21] and (ii) whether the results from the same experiments are consistent with those in [21].

Consistent results for $k \geq 2500$. The settings of the experiments in Figures 6a, 6b, 6c, and 6d are identical to those in Figures 7a, 7b, 7c, and 7d in [21], respectively. In both sets of experimental results, *SSA* and *D-SSA* outperform *IMM* when $k \geq 2500$. This confirms the findings in [21] that *SSA* and *D-SSA* are more efficient than *IMM* when k is large. In addition, in both set of experiments, *D-SSA* consistently incurs smaller running time than *SSA* does.

Inconsistent results for $k = 1$. Figure 6 shows that *IMM* is several times faster than *SSA* and *D-SSA* on all datasets when $k = 1$. Besides, Figure 7 shows that, when $k = 1$, both *SSA* and *D-SSA* produce noticeably larger numbers of RR sets than *IMM*. These results are in contradiction with those in [21], which show that *SSA* and *D-SSA* are always more efficient than *IMM* on NetHEPT, NetPHY, DBLP, and Twitter when $k = 1$. We discussed this issue with Nguyen et al. [21] and were informed that on their machine, *IMM* runs slower than *SSA* and *D-SSA*, even though the former generates a much smaller number of RR sets than the latter.

New results on Enron, Epinions, Orkut, and LiveJournal. Figures 6e-6h (resp. Figures 7e-7h) show the running time of (resp. the number of RR sets generated by) each method on Enron, Epinions, Orkut, and LiveJournal. The results are qualitatively similar to those in Figures 6a-6d and 7a-7d. That is, *IMM* is more efficient than *SSA* and *D-SSA* when $k = 1$, but the former incurs a larger running time than the latter when $k >= 2500$.

New results under the small k setting. Figure 8 shows the running time of each method under the small k setting. Interestingly, *D-SSA* is outperformed by *SSA* in most cases, which contrasts the results for the large k setting. *IMM* is more efficient than *SSA* and *D-SSA* when $k \leq 20$, but it entails a larger cost when $k \geq 30$.

New results for *SSA-Fix* and *IMM**. As shown in Figures 6-8, *SSA-Fix* offers comparable efficiency to *SSA* in all cases. This is because, although the number of random RR sets required by *SSA-Fix* is larger than that of *SSA*, the difference in the number is negligible.

Meanwhile, the figures also show that *IMM** is up to 8 times faster than *IMM*, even though they generate the same number of random RR sets. Furthermore, on *NetPHY*, *IMM** is as efficient as *SSA* and *SSA-Fix* in almost all cases. Nevertheless, the overall efficiency of *IMM** is inferior to *SSA*, *SSA-Fix*, and *D-SSA*.

5.3 Results under the LT model

Our second set of experiments evaluate the performance of each method under the LT model. Figures 9 and 10 show the computation cost of each technique under the large and small k settings, respectively; Tables 3-5 compare *SSA*, *D-SSA*, and *IMM* on their running time and the number of RR sets that they generate on Enron, Epinion, and Orkut. Note that the settings of the experiments in Figures 9a, 9b, 9c, and 9d are identical to those in Figures 6a, 6b, 6c, and 6d in [21], respectively, while the experiments in Tables 3-5 repeat the experiment in Table 3 in [21]. As in Section 5.2, we discuss our results based on whether they are new and whether they are consistent with those in [21].

Consistent results for $k \geq 2500$. As shown in Figures 9a, 9b, 9c, and 9d, *IMM* is consistently outperformed by *SSA* and *D-SSA* on when $k \geq 2500$. In addition, *D-SSA* is more efficient than *SSA*. These results are in agreement with those in [21].

Inconsistent results when $k = 1$. Figure 9 shows that *IMM* considerably outperforms *SSA* and *D-SSA* when $k = 1$ on all datasets. This contradicts the results in Figure 7 in [21], which demonstrates that *SSA* and *D-SSA* are more efficient than *IMM* on NetHEPT, NetPHY, DBLP, and Twitter when $k = 1$.

Inconsistent results in Tables 3-5. Tables 3, 4, 5 show the running time and the number of RR set of each method when $k = 1$, $k = 500$, and $k = 1000$, respectively. In addition, they include the results of the same experiments in Table 3 in [21]. (We omit *SSA-Fix* and *IMM** as they are not evaluated in [21].) Note that there exist significant differences between our results and those from [21], especially on Orkut. While the discrepancies in running time could be attributed to hardware differences, we are unable to explain why the RR set numbers that we observe differ so much from those in [21].

Nonetheless, we observe that there exist two major anomalies in the results from [21]. First, for $k = 1$ on Enron, Nguyen et al.'s results show that the numbers of RR sets generated by *D-SSA* and *SSA* are more than 40 times larger than that of *IMM*, and yet, *D-SSA* and *SSA* incur smaller computation time than *IMM* does. This is anomalous because (i) the major overheads of *D-SSA*, *SSA*, and *IMM* are incurred by the generation of RR sets, and accordingly, (ii) a significantly larger number of RR sets should indicate a higher computation overhead. One may wonder whether this anomaly is caused by the differences in how *D-SSA*, *SSA*, and *IMM* produce each RR set under the LT model. However, as we show in Appendix A, the time required by *D-SSA* and *SSA* to generate an RR set on Enron is at least 40% of that of *IMM*, which is insufficient to

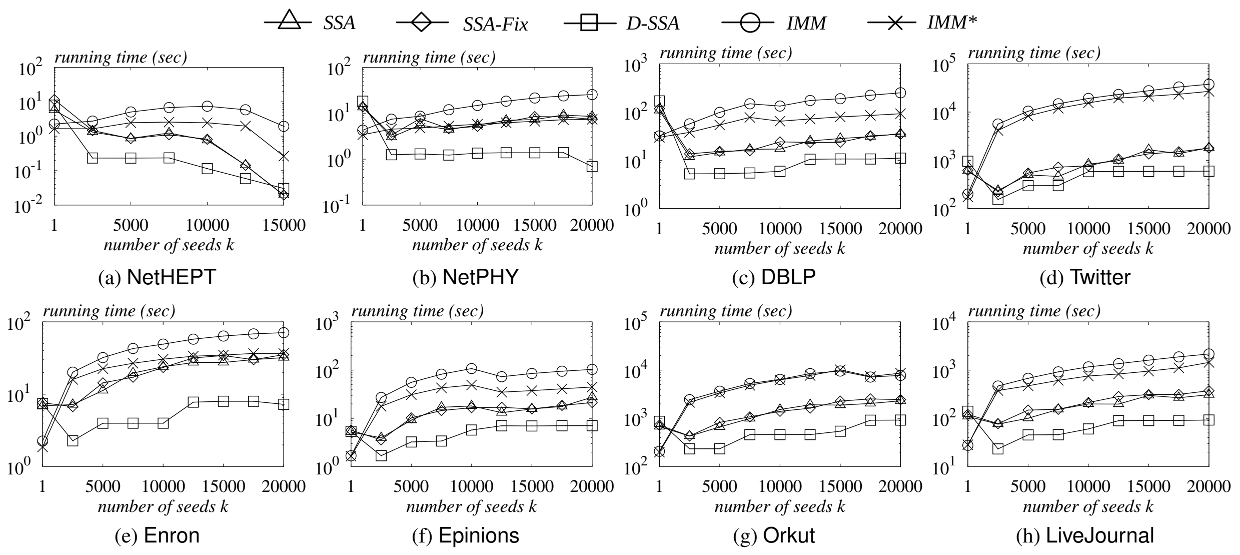


Figure 6: Running time under the IC model (large k setting).

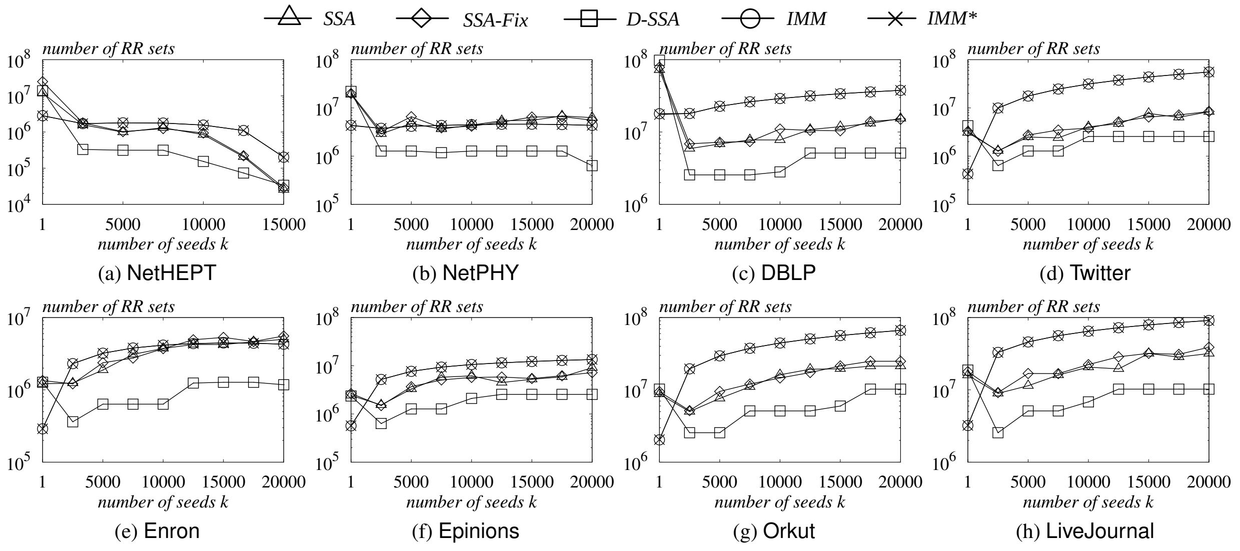


Figure 7: Number of random RR sets under the IC model (large k setting).

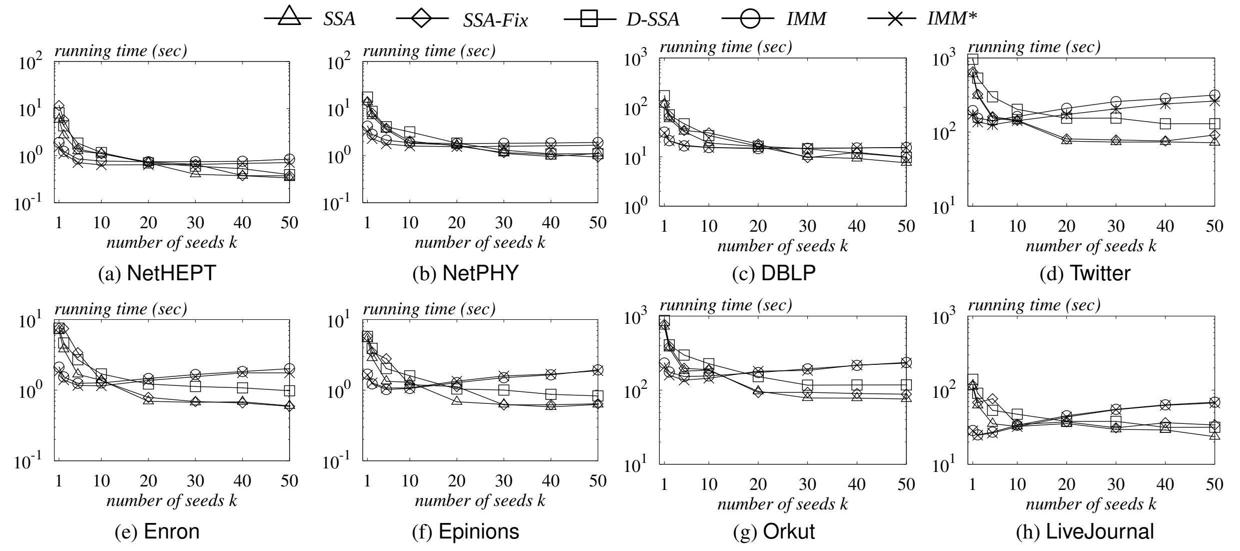
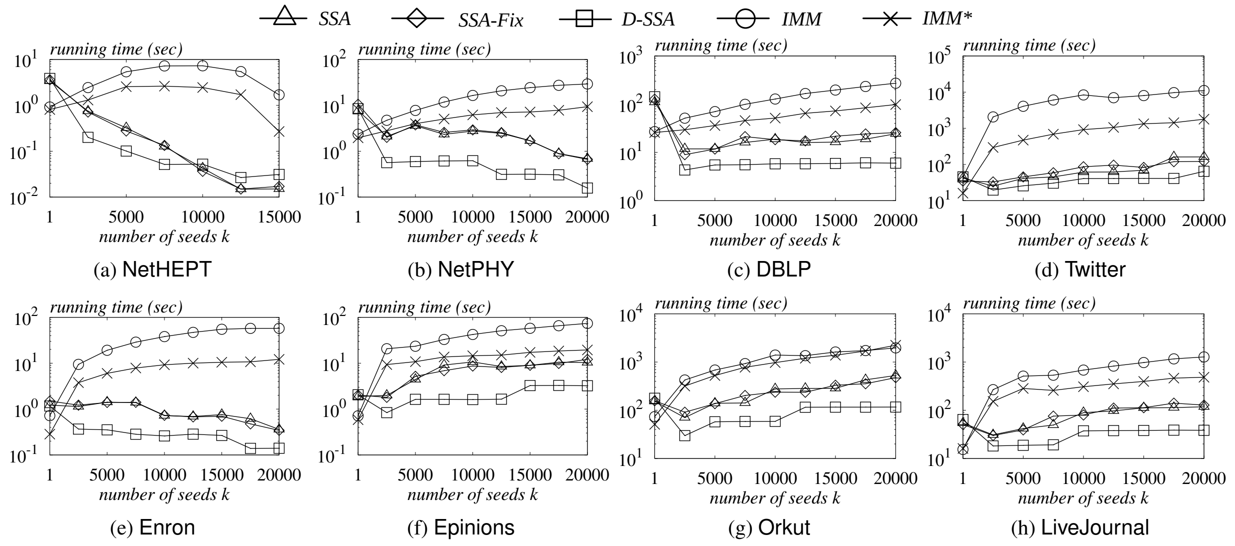


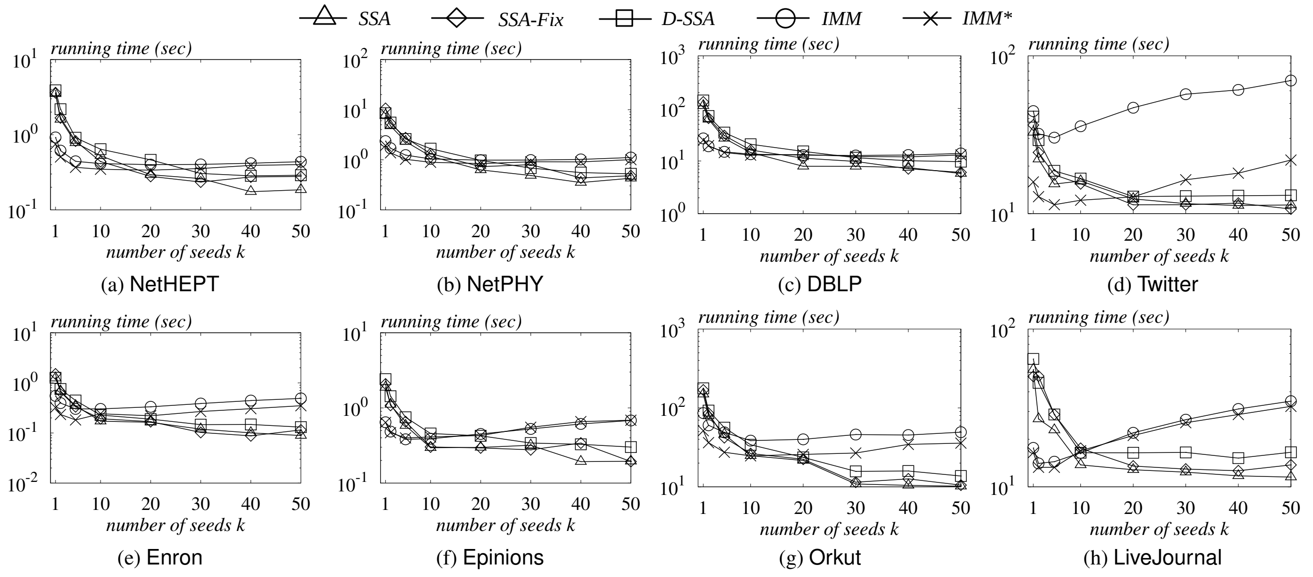
Figure 8: Running time under the IC model (small k setting).



(a) NetHEPT (b) NetPHY (c) DBLP (d) Twitter

(e) Enron (f) Epinions (g) Orkut (h) LiveJournal

Figure 9: Running time under the LT model (large k setting).



(a) NetHEPT (b) NetPHY (c) DBLP (d) Twitter

(e) Enron (f) Epinions (g) Orkut (h) LiveJournal

Figure 10: Running time under the LT model (small k setting).

offset the 40-time difference in the numbers of RR sets. In contrast, our results in Tables 3-5 do not exhibit this anomaly.

Second, Nguyen et al.'s results show that, when $k = 1$, the running time of each method on Orkut is less than 3 times of that on Enron, despite the fact that Orkut contains 82 times more nodes and 635 times more edges than Enron does. Similar issues exist for the cases of $k = 500$ and $k = 1000$. These results are irregular because the tremendous increase in the dataset size should have led to a more pronounced degradation of computation efficiency. In particular, the expected time complexity of *IMM* is $O(k(m + n)\epsilon^{-2} \log n)$ [26], which is linear to both m and n. Meanwhile, our results in Tables 3-5 show that the running time of *IMM* increases by roughly two orders of magnitude when we change the input graph from Enron to Orkut, which is much more consistent with the time complexity of *IMM*.

New results on Enron, Epinions, Orkut, and LiveJournal. Figures 9e-9h illustrate the computation overhead of each method on Enron, Epinions, Orkut, and LiveJournal. The results are largely consistent with those in Figures 9a-9d.

New results under the small *k* setting. Figure 10 illustrates the processing cost of all methods under the small k setting. The results are qualitatively similar to those in Figure 8. In particular, *IMM* is outperformed by *SSA* and *D-SSA* when $k \geq 5$, while *SSA* is generally more efficient than *D-SSA*.

New results for *SSA-Fix* and *IMM**. Figures 9 and 10 demonstrate that (i) *SSA-Fix*'s computation overhead is comparable to that of *SSA*, and (ii) *IMM** significantly outperforms *IMM* in many cases, but it is still less efficient than *SSA*, *SSA-Fix*, and *D-SSA*. These results are consistent with those under the IC model (discussed in Section 5.2).

5.4 Summary of Experimental Results

In summary, our experimental results support part of the findings in [21] that, under the IC and LT models, *SSA* and *D-SSA* are more efficient than *IMM* when k is large. In addition, we have the following new findings:

1. Contrary to the results in [21], we observe that *IMM* is considerably more efficient than *SSA* and *D-SSA* when $k = 1$, and is generally faster than the latter when $k < 20$.

Table 3: Number of RR sets under the LT model with $k = 1$.

Dataset	Source	RR set num. ($\times 10^3$)			Running time (sec)		
		D-SSA	*SSA*	*IMM*	*D-SSA*	*SSA*	*IMM*
Enron	ours	954	1073	211	1.26	1.22	0.62
	[21]	120	180	2.8	0.2	0.6	0.7
Epinion	ours	2234	2149	489	2.3	1.8	0.7
	[21]	160	300	400	0.4	0.8	0.8
Orkut	ours	5115	4678	1177	178.2	152.7	69.4
	[21]	30	60	124	0.5	1.1	2.1

Table 4: Number of RR sets under the LT model with $k = 500$.

Dataset	Source	RR set num. ($\times 10^3$)			Running time (sec)		
		D-SSA	*SSA*	*IMM*	*D-SSA*	*SSA*	*IMM*
Enron	ours	82	176	579	0.15	0.25	2.5
	[21]	30	60	580	0.1	0.2	3.1
Epinion	ours	250	378	1409	0.36	0.44	3.5
	[21]	40	60	1214	0.1	0.1	4.4
Orkut	ours	315	473	2135	14.6	18.9	140.0
	[21]	60	240	760	0.6	3.4	17.4

Table 5: Number of RR sets under the LT model with $k = 1000$.

Dataset	Source	RR set num. ($\times 10^3$)			Running time (sec)		
		D-SSA	*SSA*	*IMM*	*D-SSA*	*SSA*	*IMM*
Enron	ours	154	333	911	0.30	0.59	4.4
	[21]	120	180	910	0.2	0.6	6.9
Epinion	ours	314	644	2226	0.46	0.86	8.2
	[21]	160	240	1852	0.3	0.7	12.1
Orkut	ours	422	878	3279	27.2	39.1	263.0
	[21]	60	240	1392	0.6	4.3	45.5

2. When $k \leq 50$, *SSA* is at least as efficient as or more efficient than *D-SSA* in most cases.

3. *SSA-Fix* offers comparable performance to *SSA* in all cases tested.

4. The performance gap between *IMM* and *SSA* (resp. *D-SSA*) is partially due to the fact that the implementation of the former is less optimized than that of *SSA* (resp. *D-SSA*). We show that an optimized implementation of *IMM* (i.e., *IMM**) can be up to an order of magnitude faster than the original implementation, but its overall efficiency is still inferior to that of *SSA* and *D-SSA*.

Remark. We have also tested the expected influence of the seed sets generated by each algorithm. Our results show that, given the same experimental setting, all methods achieve similar expected influence, i.e., there is no significant difference in the quality of results returned by each method. In addition, the expected influence of each method increases considerably from when k increases 1 to 2500, but does not increase much when k further increases from 2500 to 20000. This indicates that $k < 2500$ is likely to be a more realistic setting for the influence propagation models that we have tested. Due to the space constraint, we omit the results from this paper but include them in our technical report [1].

6. CONCLUSION

This paper revisits *SSA* and *D-SSA* [21], which are two recently proposed algorithms for influence maximization that claim to offer non-trivial accuracy guarantees, optimal time efficiency, and strong empirical performance. We carefully analyze the key lemmas and theorems in [21] and conduct a comprehensive set of experiments to re-evaluate *SSA* and *D-SSA* against a competing method, *IMM* [26]. Our analysis shows that the accuracy and efficiency claims of both *SSA* and *D-SSA* are flawed, and we provide a revised version of *SSA*, referred to as *SSA-Fix*, which restores the approximation guarantee of *SSA*. Meanwhile, our experiments confirm the finding in [21] that *SSA* and *D-SSA* are more efficient than *IMM* when the size k of the seed set is large, but we also find that, contrary to the results reported in [21], *IMM* generally outperforms *SSA* and *D-SSA* when k is small. In addition, we observe that there exist several other anomalies in the experimental results in [21], and that *SSA-Fix* is largely as efficient as *SSA*. Our study leaves open the question of whether one can improve *SSA* (resp. *D-SSA*) to fix both its accuracy and efficiency claims and to reduce its empirical overheads when k is small, e.g., $k <= 1000$.

7. ACKNOWLEDGEMENT

Research by Huang, Wang, and Xiao was supported in part by grant MOE2015-T2-2-069 from MOE, Singapore. Research by Bevilacqua was supported by a Canada Graduate Doctoral Scholarship from NSERC, while research by Lakshmanan was supported in part by a grant from NSERC.

8. REFERENCES

[1] `https://sites.google.com/site/vldb2017imexptr/`.

[2] `https://sourceforge.net/projects/im-imm/`.

[3] `https://github.com/hungnt55/Stop-and-Stare`.

[4] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *WINE*, pages 306–311, 2007.

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

[6] W. Chen, W. Lu, and N. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *AAAI*, 2012.

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

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

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

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

[11] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

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

[13] M. Gomez-Rodriguez and B. Schölkopf. Influence maximization in continuous time diffusion networks. In *ICML*, 2012.

[14] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A data-based approach to social influence maximization. *PVLDB*, 5(1):73–84, 2011.

[15] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, pages 47–48, 2011.

[16] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *ICDM*, pages 918–923, 2012.

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

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

Table 6: Average time required to generate a random RR set under the LT model.

Method	Time (μs)							
	NetHEPT	NetPHY	DBLP	Twitter	Enron	Epinions	Orkut	LiveJournal
IMM	0.83	0.93	1.89	96.20	2.51	1.32	48.50	5.50
SSA, *SSA-Fix*, *D-SSA*	0.76	0.79	1.37	9.89	1.10	1.06	23.93	4.40

[19] J. Kim, S.-K. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks. In *ICDE*, pages 266–277, 2013.

[20] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

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

[22] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. *CoRR*, abs/1605.07990v2, Sep 7, 2016.

[23] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. *CoRR*, abs/1605.07990v3, Feb 22, 2017.

[24] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.

[25] L. Seeman and Y. Singer. Adaptive seeding in social networks. In *FOCS*, pages 459–468, 2013.

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

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

[28] V. V. Vazirani. *Approximation Algorithms*. Springer, 2002.

[29] X. Wang, Y. Zhang, W. Zhang, X. Lin, and C. Chen. Bring order into the samples: A novel scalable method for influence maximization. *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[30] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *KDD*, pages 1039–1048, 2010.

APPENDIX

A. GENERATION OF RR SETS UNDER THE LINEAR THRESHOLD MODEL

As with the IC model, the *linear threshold (LT)* model [17] also assumes that each edge e in the social network G is associated with a propagation probability $p(e)$, but it requires that for each node v in G, the propagation probabilities of all edges ending to v sum up to no more than 1. In addition, given a seed set S, it considers a different type of influence propagation process as follows:

1. At timestamp 0, for each node v in G, we assign a threshold $\theta(v)$ that is sampled uniformly at random from $[0, 1]$. Then, we activate the nodes in S.
2. At timestamp i, for each inactive node w, we examine all of its incoming edges from activated nodes and check if the sum of those edges' propagation probabilities is not smaller than $\theta(v)$. If it is, then we activate w; otherwise, w remains inactive.
3. The influence propagation process terminates when none of the remaining inactivate nodes can be activated.

As shown in [27], a random RR set R under the LT model can be generated using an iterative procedure. In the first iteration, we choose a node u uniformly at random from G and set $R = \{u\}$. Then, in the i-th ($i > 1$) iteration, we examine the nodes inserted into R in the $(i-1)$-th iteration. For each v of those nodes, we identify the set $E_{in}(v)$ of edges that end at v, and we toss a coin that lands heads with a probability of $\sum_{e \in E_{in}(v)} p(e)$. If the coin shows a head, then we sample an edge e' from $E_{in}(v)$, such that the sampling probability of each edge is proportional to its propagation probability; after that, we insert into R the node that e' starts from (if the node is not already in R). On the other hand, if the coin shows a tail, then we do not perform any operation for v. This iterative procedure terminates when no node is inserted into R in a certain iteration.

IMM, *SSA*, *SSA-Fix*, and *D-SSA* all implement the above iterative procedure to generate random RR sets under the LT model but their implementations differ in the way that they sample an edge from each edge set $E_{in}(v)$. Without loss of generality, assume that $E_{in}(v)$ contains α edges $e_1, e_2, \ldots, e_\alpha$. Given $E_{in}(v)$, *IMM* first samples a number x uniformly at random from $[0, 1]$, after which it performs a linear scan on $e_1, e_2, \ldots, e_\alpha$. If $x > \sum_{i=1}^{\alpha} p(e_i)$, then *IMM* omits all edges in $E_{in}(v)$; otherwise, it retrieves e_β, where β is the smallest integer satisfying $\sum_{i=1}^{\beta} p(e_i) \geq x$. Observe that this edge sampling process takes $O(\alpha)$ time.

In contrast, *SSA*, *SSA-Fix*, and *D-SSA* only take $O(\log \alpha)$ time to generate e_β. In particular, for each node v in G, they preprocess $E_{in}(v)$ to obtain the prefix sum of $p(e_1), p(e_2), \ldots, p(e_\alpha)$, namely, they pre-compute an α-element array for v such that the j-th element equals $\sum_{i=1}^{j} p(e_i)$. After that, whenever they need to sample an edge from $E_{in}(v)$, they first generate a number x uniformly at random from $[0, 1]$, and then they perform a binary search on the prefix sum for v to identify the smallest integer β with $\sum_{i=1}^{\beta} p(e_i) \geq x$. If β does not exists (i.e., $x > \sum_{i=1}^{\alpha} p(e_i)$), then they omit all edges in $E_{in}(v)$; otherwise, they use e_β as the edge sampled from $E_{in}(v)$.

To illustrate the efficiency difference between the above two edge sampling approaches, we measure the time required by *IMM*, *SSA*, *SSA-Fix*, and *D-SSA* to generate 10^6 random RR sets on each dataset, and we report the average time per RR set in Table 6. Observe that the time required by *IMM* is up to 10 times larger than that by the other algorithms.

In addition to using an optimized edge sampling procedure, *SSA*, *SSA-Fix*, and *D-SSA* incorporate another trick to reduce the overhead of RR set generation. In particular, in Line 7 of *SSA* (i.e., Algorithm 1), it invokes Algorithm 3 to estimate the expected spread of a seed set S_k, in which it generates random RR sets one by one and checks if they intersect S_k. For each RR set R involved in this process, *SSA* does not wait until R is full constructed to check if R and S_k overlap. Instead, when *SSA* constructs R, it examines each node v being inserted into R, and checks whether it is in S_k; if $v \in S_k$, then *SSA* immediately terminates the generation of R since it is clear that R intersects S_k. The same trick is also adopted in *SSA-Fix* and Lines 5-6 of *D-SSA* (i.e., Algorithm 4).