Revisiting of 'Revisiting the Stop-and-Stare Algorithms for Influence Maximization'

Hung T. Nguyen[1,2], Thang N. Dinh[1], and My T. Thai[3]

[1] Virginia Commonwealth University, Richmond VA 23284, USA
{hungnt, tndinh}@vcu.edu
[2] Carnegie Mellon University, Pittsburgh, PA 15213, USA
[3] University of Florida, Gainesville FL 32611, USA
mythai@cise.ufl.edu

Abstract. SSA/DSSA were introduced in SIGMOD'16 as the first algorithms that can provide rigorous $1-1/e-\epsilon$ guarantee with fewer samples than the worst-case sample complexity $O(nk\frac{\log n}{\epsilon^2 OPT_k})$. They are order of magnitude faster than the existing methods. The orginal SIGMOD'16 paper, however, contains errors, and the new fixes for SSA/DSSA, referred to as SSA-fix and D-SSA-fix, have been published in the extended version of the paper [11]. In this paper, we affirm the correctness on accuracy and efficiency of SSA-fix/D-SSA-fix algorithms. Specifically, we refuse the misclaims on 'important gaps' in the proof of D-SSA-fix's efficiency raised by Huang et al. [5] published in VLDB in May 2017. We also replicate the experiments to dispute the experimental discrepancies shown in [5]. Our experiment results indicate that implementation/modification details and data pre-processing attribute for most discrepancies in running-time. [4]

Keywords: Influence Maximization · Stop-and-Stare · Approximation Algorithm.

1 Introduction

Given a network $G = (V, E)$ and an integer k, the *influence maximization* (IM) asks for a subset of k nodes, called seed set, that can influence maximum number of nodes in the network under a diffusion model. The problem has produced a long line of research results, e.g., those in [6, 3, 1, 16, 11] and references therein.

RIS framework. A key breakthrough for the problem is the introduction of a novel technique, called *reverse influence sampling* (RIS), by Borgs et al. [1]. The RIS framework, followed by all works discussed in this paper [16, 11, 5], will

- Generate a collection $\{R_1, R_2, \ldots, R_T\}$ of *Reversed Reachability Sets* (or RR sets). Each RR set R_i is generated by selecting a random node u and perform

[4] We requested the modified code from VLDB '17 [5] last year but have not received the code from the authors. We also sent them the explanation for the gaps they misclaimed for the D-SSA-fix's efficency proof but have not recieved their concrete feedback.

a reversed traversal from u to include into R_i all nodes that can reach to u, i.e., can influence u.
- Find a subset S of k nodes that can cover a maximum number of RR sets using the greedy algorithm for the maximum coverage problem.
- The returned solution S will be a $(1 - 1/e - \epsilon)$ solution, for large T.

Worst-case Sample Complexity. The sample complexity, i.e., the number of RR sets to guarantee a $(1 - 1/e - \epsilon)$ approximation factor is shown to be $\theta(k, \epsilon) = O(nk\frac{\log n}{\epsilon^2 \mathsf{OPT}_k})$ [17] where OPT_k denotes the expected influence of an optimal solution. Unfortunately, $\theta(k, \epsilon)$ depends on OPT_k, an unknown, thus, it is challenging to know whether or not $\theta(k, \epsilon)$ samples have been generated.

Tang et al. [16] proposed IMM algorithm that stops when $\theta(k, \epsilon)$ samples have been generated. Recently, a flaw in the analysis of IMM has been pointed out by Wei Chen [2] together with a fix for IMM. Independently, we proposed in [13] BCT, an algorithm that also stops within $O(\theta(k, \epsilon))$ samples for generalized versions of IM, with heterogeneous cost and influence effect.

Unfortunately, even meeting the sample complexity $\theta(k, \epsilon)$ is not efficient enough for billion-scale networks. In several weighted models, such as Trivalency or constant probability [11, 14, 15], IMM (and BCT) struggles for the largest test networks such as Twitter and Friendster datasets. The main reason is that $\theta(k, \epsilon)$ is a **worst-case** sample complexity, thus, it is *very conservative* in practice. The θ threshold needs to hold for all "hard" inputs, which rarely happens in practice. Can we achieve $(1 - 1/e - \epsilon)$ approximation guarantee with fewer than $\theta(k, \epsilon)$ samples?

Stop-and-Stare and Instance-specific Sample Complexity. SSA and D-SSA were introduced in our SIGMOD'16 [8] as the first algorithms that can guarantee $(1 - 1/e - \epsilon)$ optimality with fewer than $\theta(k, \epsilon)$ samples. For each specific instance $\Pi = (G = (V, E), k, \epsilon)$ of IM, SSA and D-SSA aim to reduce the sample complexity to some instance-specific thresholds. Unlike the worst-case threshold θ, instance-specific thresholds adapt to the actual complexity of the input including the information contained in network structure and influence landscape. Thus, those thresholds can be several orders of magnitude smaller than θ, especially, for 'easy' instances of IM. Consequently, algorithms that meet this new thresholds are potentially 1,000 times (or more) faster than IMM [16] and BCT [13].

Specifically, SSA and D-SSA were designed to provide $1 - 1/e - \epsilon$ guarantees using only $O(N^{(1)}_{min}$ and $O(N^{(2)}_{min}$ samples where $N^{(1)}_{min} > N^{(2)}_{min}$, termed Type-1 and Type-2 minimum thresholds [8], respectively. $N^{(1)}_{min}$ and $N^{(2)}_{min}$ are instance-specific lower-bounds on the number of necessary samples and can be many times smaller than $\theta(\epsilon, k)$ in practice. Specifically, they are the lower-bounds for IM's algorithms following "out-of-sample validation" approaches that: 1) continuously, generating two pools (of increasingly sizes) of samples, one for finding a candidate solution and one for 'validating' the candidate solution; and 2) stop when the discrepancies in the estimations of the candiate in the two pools are sufficiently small. Unlike $\theta(\epsilon, k)$, our new lower bounds $N^{(1)}_{min}$ and $N^{(2)}_{min}$ still vary widely among inputs that share the parameters n, k, and OPT_k.

Table 1: Summary of papers related to SSA and D-SSA algorithms

Papers	Date	Contribution
Nguyen et al. SIGMOD'16 [8]	25 May 2016	Stop-and-Stare algorithms SSA/D-SSA proposed.
Personal communication	Jul. 2016	We identified the issues in the proof of SSA/D-SSA thanks to anonymous reviewers for IEEE/ACM ToN. The reviewers pointed out a similar mistake in our submitted manuscript [10].
Nguyen et al. ArXiv-v2 [9]	7 Sep 2016	SSA-fix provided (D-SSA remained broken)
Huang et al. VLDB [5] early version	15 Jan. 2017	Identified the errors in proofs for approximation factor and sample efficiency for SSA and D-SSA and provided a similar SSA-fix
Nguyen et al. ArXiv-v3 [11]	22 Feb 2017	Provided D-SSA-fix and correct proofs for approximation factor and sample efficiency for SSA-fix and D-SSA-fix
Huang et al. VLDB'17 [5]	May 2017	Adding a claim on the flaw in the proof for D-SSA-fix's sample efficiency in [11]. No concerns raised for the proof on SSA-fix/D-SSA-fix approximability and SSA-fix's sample efficiency.
This paper	Oct. 2018	Affirmed the D-SSA-fix's sample efficiency, rejecting the doubt raised in Huang et al. [5]

In summary, the key contribution in [11] is that SSA/D-SSA are not only $(1 - 1/e - \epsilon)$ approximation algorithms but also are asymptotically optimal in terms of the proposed instance-specific sample complexities.

Our work in [8] consists of 4 major proofs:

- **SSA's and D-SSA's approximability**: showing that SSA and D-SSA return $1 - 1/e - \epsilon$ solutions with high probability (2 proofs).
- **SSA's and D-SSA's efficiency**: showing that SSA and D-SSA using only cT_1 and cT_2 samples where $\theta(k, \epsilon) \gg T_1 > T_2$ are instance-specific sample complexities and c is a fixed constant (2 proofs).

Errors in our proofs [8] and fixes. Our proofs contain flaws which comes from the *applying of concentration inequalities in which the parameters, such as ϵ, δ and the number of samples, may depend on the generated samples.*

The errors were brought to our attention through two channels: 1) the same flaw pointed out by anonymous reviewers for one of our submission (not the published version) to IEEE/ACM Transaction to Networking [10] in Jul. 2016 and 2) an early manuscript of Huang et al. [5] in Jan. 2017 and concerns on the martingales sent to us by the authors of [5].

Upon discovering the errors, we uploaded the fix for SSA, called SSA-fix, on Arxiv on Sep. 2016 [9] and the fix for D-SSA, called D-SSA-fix, with corrected proofs in Feb. 2017 [11].

The final version of Huang et al. [5], while not giving any comments on the 3 proofs for SSA-fix's approximability, SSA-fix's efficiency, and D-SSA-fix's approximability, claims *"important gaps" in our proof for D-SSA-fix's efficiency.* While we appreciated the errors pointed out in Huang et al. [5] for our original paper in SIGMOD'16 [8], we found the claim on "important gaps" of D-SSA-fix's efficiency is a misclaim.

This paper aims to affirm the correctness of D-SSA-fix's efficiency in [11], explaining the 'important gaps' claimed in Huang et al.[5] (and their extended version [4]) and explain the discrepancies in experiments claimed by Huang et al.[5]. We summarize the timeline of publication and correspondence in Table 1.

Organization. We first summarize our fixes for SSA/DSSA in our Arxiv [11]. Then we provide justification for the claimed by [5] on the "important gaps" for D-SSA-fix's efficiency. Finally, we present the experiment to explain the observed discrepancies in [5].

Algorithm 1: SSA-fix

Input: Graph G, $0 \leq \epsilon, \delta \leq 1$, and a budget k
Output: An $(1 - 1/e - \epsilon)$-optimal solution, $\hat{S}_k$ with at least $(1 - \delta)$-probability

Choose $\epsilon_1, \epsilon_2, \epsilon_3$ satisfying Equation 18 in [11];
$N_{max} = 8\frac{1-1/e}{2+2\epsilon/3}\Upsilon\left(\epsilon, \frac{\delta}{6}/\binom{n}{k}\right)\frac{n}{k}$; $i_{max} = \lceil \log_2 \frac{2N_{max}}{\Upsilon(\epsilon_3, \delta/3)} \rceil$;
$\Lambda_1 \leftarrow (1+\epsilon_1)(1+\epsilon_2)\Upsilon(\epsilon_3, \frac{\delta}{3i_{max}})$
$\mathcal{R} \leftarrow$ Generate Λ_1 random RR sets
repeat
    Double the size of $\mathcal{R}$ with new random RR sets
    $<\hat{S}_k, \hat{\mathbb{I}}(\hat{S}_k)> \leftarrow$ Max-Coverage($\mathcal{R}$, k, n)
    if $\text{Cov}_{\mathcal{R}}(\hat{S}_k) \geq \Lambda_1$ **then** ▷ `Condition C1`
        $\delta_2' = \frac{\delta_2}{3i_{max}}$; $T_{max} = 2|\mathcal{R}|\frac{1+\epsilon_2}{1-\epsilon_2}\frac{\epsilon_3^2}{\epsilon_2^2}$
        $\mathbb{I}_c(\hat{S}_k) \leftarrow$ Estimate-Inf$(G, \hat{S}_k, \epsilon_2, \delta_2', T_{max})$
        if $\hat{\mathbb{I}}(\hat{S}_k) \leq (1+\epsilon_1)\mathbb{I}_c(\hat{S}_k)$ **then** ▷ `Condition C2`
            return $\hat{S}_k$
until $|\mathcal{R}| \geq N_{max}$;
return $\hat{S}_k$

Algorithm 2: The original D-SSA algorithm [8]

Input: Graph G, $0 \leq \epsilon, \delta \leq 1$, and k
Output: An $(1 - 1/e - \epsilon)$-optimal solution, $\hat{S}_k$

$\Lambda \leftarrow 2c(1+\epsilon)^2 \log(\frac{2}{\delta})\frac{1}{\epsilon^2}$
$\mathcal{R} \leftarrow$ Generate Λ random RR sets by RIS
$<\hat{S}_k, \hat{\mathbb{I}}(\hat{S}_k)> \leftarrow$ Max-Coverage$(\mathcal{R}, k)$
repeat
  $\mathcal{R}' \leftarrow$ Generate $|\mathcal{R}|$ random RR sets by RIS
  $\mathbb{I}_c(\hat{S}_k) \leftarrow \text{Cov}_{\mathcal{R}'}(\hat{S}_k) \cdot n/|\mathcal{R}'|$
  $\epsilon_1 \leftarrow \hat{\mathbb{I}}(\hat{S}_k)/\mathbb{I}_c(\hat{S}_k) - 1$
  if $(\epsilon_1 \leq \epsilon)$ **then**
    $\epsilon_2 \leftarrow \frac{\epsilon-\epsilon_1}{2(1+\epsilon_1)}$, $\epsilon_3 \leftarrow \frac{\epsilon-\epsilon_1}{2(1-1/e)}$
    $\delta_1 \leftarrow e^{-\frac{\text{Cov}_{\mathcal{R}}(\hat{S}_k)\cdot\epsilon_3^2}{2c(1+\epsilon_1)(1+\epsilon_2)}}$
    $\delta_2 \leftarrow e^{-\frac{(\text{Cov}_{\mathcal{R}'}(\hat{S}_k)-1)\cdot\epsilon_2^2}{2c(1+\epsilon_2)}}$
    if $\delta_1 + \delta_2 \leq \delta$ **then**
      return $\hat{S}_k$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$
  $<\hat{S}_k, \hat{\mathbb{I}}(\hat{S}_k)> \leftarrow$ Max-Coverage$(\mathcal{R}, k)$
until $|\mathcal{R}| \geq (8+2\epsilon)n\frac{\ln\frac{2}{\delta}+\ln\binom{n}{k}}{k\epsilon^2}$;
return $\hat{S}_k$

Algorithm 3: D-SSA-fix

Input: Graph G, $0 \leq \epsilon, \delta \leq 1$, and k
Output: An $(1 - 1/e - \epsilon)$-optimal solution, $\hat{S}_k$

$N_{max} = 8\frac{1-1/e}{2+2\epsilon/3}\Upsilon\left(\epsilon, \frac{\delta}{6}/\binom{n}{k}\right)\frac{n}{k}$;
$t_{max} = \lceil \log_2(2N_{max}/\Upsilon(\epsilon, \frac{\delta}{3})) \rceil$; $t = 0$;
$\Lambda_1 = 1 + (1+\epsilon)\Upsilon(\epsilon, \frac{\delta}{3t_{max}})$;
repeat
  $t \leftarrow t + 1$;
  $\mathcal{R}_t = \{R_1, \ldots, R_{\Lambda_1 2^{t-1}}\}$;
  $\mathcal{R}_t^c = \{R_{\Lambda_1 2^{t-1}+1}, \ldots, R_{\Lambda_1 2^t}\}$;
  $< \hat{S}_k, \hat{\mathbb{I}}_t(\hat{S}_k) > \leftarrow$ Max-Coverage$(\mathcal{R}_t, k)$;
  if $\text{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \geq \Lambda_1$ **then** ▷ Condition D1
    $\mathbb{I}_t^c(\hat{S}_k) \leftarrow \text{Cov}_{\mathcal{R}_t^c}(\hat{S}_k) \cdot n/|\mathcal{R}_t^c|$;
    $\epsilon_1 \leftarrow \hat{\mathbb{I}}_t(\hat{S}_k)/\mathbb{I}_t^c(\hat{S}_k) - 1$;
    $\epsilon_2 \leftarrow \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}}$; $\epsilon_3 \leftarrow \epsilon\sqrt{\frac{n(1+\epsilon)(1-1/e-\epsilon)}{(1+\epsilon/3)2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}}$;
    $\epsilon_t = (\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2)(1 - 1/e - \epsilon) + (1 - \frac{1}{e})\epsilon_3$;
    if $\epsilon_t \leq \epsilon$ **then** ▷ Condition D2
      return $\hat{S}_k$;
until $|\mathcal{R}_t| \geq N_{max}$;
return $\hat{S}_k$;

2 Summary of SSA-fix and D-SSA-fix [11]

First, we summarize the errors and our fixes for SSA/D-SSA algorithms and refer to the extended version of our SIGMOD paper in [11] for complete proofs.

2.1 Fixes for SSA algorithm

The main idea of SSA (and the Stop-and-Stare framework) is to 1) generate a collection of samples $\mathcal{R}$ and find a candidate solution $\hat{S}_k$ using the greedy algorithm; 2) measure the difference between (biased) influence of $\hat{S}_k$ with samples in $\mathcal{R}$ and an unbiased estimation of $\hat{S}_k$ on another set of samples; and 3) the algorithm stops if the difference is small enough, otherwise, it doubles the number of generated samples.

Summary of errors for SSA. The influence for candidate solution $\hat{S}_k$ is estimated multiple times. And the error probability did not take this fact into the account.

Summary of changes in SSA-fix. As highlighted in Algorithm 1, to account for the multiple influence estimates by Estimate-Inf procedure, we decrease the error probability by a factor $i_{max} = O(\log n)$. Specifically, Algorithm 1 introduces the factor i_{max} and divides the probability guarantee δ_2 by i_{max} in Lines 2 and 9. Through union bound, we can show that this sufficiently accounts for the cumulative error in Estimate-Inf while insignificantly affecting the number of samples.

Note that [5] provides the same fix by decreasing the error probability by a factor $O(\log n)$.

2.2 Fixes for D-SSAalgorithm

Summary of errors for D-SSA. In the orginial D-SSA, presented in Algorithm 2, the computations of δ_1 and δ_2 depend on ϵ_1, ϵ_2 and ϵ_3, which, in turn, depend on the generated samples. This dependency on the generated samples make the proof incorrect as the of Chernoff's inequality.

Summary of changes in D-SSA-fix. Our D-SSA-fix, shown in Algorithm 3, set $\delta_1 = \delta_2 = c'\delta$ for a fixed constant c'. The Chernoff's bounds are applied to bound the errors ϵ_1 and ϵ_2 at the fixed points when the number of samples are $\Lambda_1 2^i$, for $i = 1, 2, \ldots, \lceil \log N_{max} \rceil$. This change is reflected in the Lines 9-14.

We compute ϵ_1, the discrepancy of estimating using two different collections of RR sets, i.e. $\mathcal{R}$ and $\mathcal{R}'$; ϵ_2 and ϵ_3 bound the maximum estimation errors with high probability. At a first glance, ϵ_2 and ϵ_3 still seem to depend on the generated RR sets in $\mathcal{R}_t^c$. However, $\frac{\mathbb{I}_t^c(\hat{S}_k)}{1+\epsilon}$ serves as a lower-bound for $\mathbb{I}(\hat{S}_k)$ with high probability and can be used in the bounding of ϵ_2 and ϵ_3.

3 Affirming the correctness in D-SSA-fix's efficiency [11]

The final version of Huang et al. [5] claims *"important gaps" in our proof for D-SSA-fix's efficiency*. Here we provide the details showing this misclaim and affirm the correctness for D-SSA-fix's efficiency proof.

3.1 Gap in showing $\epsilon_2 \leq \epsilon_0/3$ and explanation

Gap claimed by Huang et al. [5, 4]. The first gap claimed by Huang et al. [4] (Page 14, B.1 Misclaim) is about Eqs. (93) and (94), in the proof of Theorem 6, in [11]. Below we quote those two equations in [11].

> Apply the inequalities $2^{t-1} \geq \alpha \frac{n}{\mathsf{OPT}_k}$ [a], Eq. (88), and $\mathbb{I}(\hat{S}_k) \geq (1 - 1/e - \epsilon)\mathsf{OPT}_k$, Eq. (87). For **sufficiently large** $\alpha > \frac{\mathbf{9(1+\epsilon)}}{\mathbf{(1-1/e-\epsilon)}}$, we have
>
> $$\epsilon_2 = \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}} \leq \epsilon_0/3 \leq \epsilon_b^*/3 \quad (93)$$
>
> $$\epsilon_3 = \epsilon\sqrt{\frac{n(1+\epsilon)(1-1/e-\epsilon)}{(1+\epsilon/3)2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}} \leq \epsilon_0/3 \leq \epsilon_b^*/3 \quad (94)$$
>
> [a] The Eq. (88) in [11] states $2^{t-1} \geq \alpha \frac{n}{\mathsf{OPT}_k}\frac{\epsilon^2}{\epsilon_0^2}$. Here, the factor $\frac{\epsilon^2}{\epsilon_0^2}$ were missed due to a typo.

Figure 1.1: Proof of Theorem 6 in [11]. Huang et al. [4] claimed important gaps for Eqs. (93) and (94).

Huang et al. raised the concern that their derivation using Eqs. (87) and (88) do not lead to Eqs. (93) and (94).

Our response. The ommitted detail is that $\mathbb{I}_t^c(\hat{S}_k)$ is an unbiased estimator of $\mathbb{I}(\hat{S}_k)$, thus, its value concentrates around $\mathbb{I}(\hat{S}_k)$. In fact, from Eq. (89), $\tilde{\epsilon_t} \leq \frac{\epsilon_0}{3}$, and Eq. (90), $\hat{\mathbb{I}}_t^c(\hat{S}_k) \geq (1 - \tilde{\epsilon}_t)\mathbb{I}(\hat{S}_k)$, it follows that

$$\hat{\mathbb{I}}_t^c(\hat{S}_k) \geq (1 - \frac{\epsilon_0}{3})\mathbb{I}(\hat{S}_k) \geq (1 - \frac{\epsilon}{3})\mathbb{I}(\hat{S}_k) \quad (\text{since } \epsilon_0 = \min\{\epsilon, \epsilon_b^*)\}) \quad (1)$$

Thus, picking a sufficiently large constant $\alpha = \frac{100(1+\epsilon)}{(1-1/e-\epsilon)}$, we have

$$\epsilon_2 = \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}\mathbb{I}_t^c(\hat{S}_k)}} \leq \epsilon\sqrt{\frac{n(1+\epsilon)}{2^{t-1}(1-\frac{\epsilon}{3})\mathbb{I}(\hat{S}_k)}} \leq \epsilon\sqrt{\frac{n(1+\epsilon)}{\alpha\frac{n}{\mathsf{OPT}_k}\frac{\epsilon^2}{\epsilon_0^2}(1-\frac{\epsilon}{3})\mathbb{I}(\hat{S}_k)}} (\text{by Eq. 88})$$

$$< \frac{\epsilon_0}{3}\sqrt{\frac{(1-1/e-\epsilon)\mathsf{OPT}_k}{\mathbb{I}(\hat{S}_k)}} \leq \frac{\epsilon_0}{3} \text{ (by Eq. 87)} \quad (93)$$

Since $\epsilon_3 < \epsilon_2$, we also have $\epsilon_3 \leq \epsilon_0/3 \leq \epsilon_b^*/3$, i.e., Eq. (94) follows.

...
In addition, the proof assumes that we can set the constant α to be larger than $\frac{9(1+\epsilon)}{(1-1/e-\epsilon)}$, but this assumption does not always hold. In particular, as shown in Equation 27, α must ensure that $2^{t-1} \geq \alpha \frac{n}{\mathsf{OPT}_k} \frac{\epsilon^2}{\epsilon_0^2}$, which leads to

$$\alpha \leq \frac{2^{t-1}\mathsf{OPT}_k \epsilon_0^2}{n\epsilon^2}. \tag{2}$$

Therefore, we can set $\alpha \geq \frac{9(1+\epsilon)}{(1-1/e-\epsilon)}$, only if

$$\frac{9(1+\epsilon)}{(1-1/e-\epsilon)} \leq \frac{2^{t-1}\mathsf{OPT}_k \epsilon_0^2}{n\epsilon^2}. \tag{3}$$

This inequality is never established in the proof.

Figure 1.2: Argument in [4] on the gap of setting constant α

3.2 Gap on setting of α and explanation.

Gap claimed by Huang et al. [4]. The second gap is shown in Figure 1.2. Huang et al. raised the concern that the constant α may not exist due to bounded range between $\frac{9(1+\epsilon)}{(1-1/e-\epsilon)}$ and $\frac{2^{t-1}\mathsf{OPT}_k \epsilon_0^2}{n\epsilon^2}$.
Our response. We first select a fixed and sufficiently large constant α, e.g., setting $\alpha = \frac{100(1+\epsilon)}{(\mathbf{1-1/e-\epsilon})}$. There is no need to choose α to satify the inequality $2^{t-1} \geq \alpha \frac{n}{\mathsf{OPT}_k}$. Indeed, for a fixed constant α at some sufficiently large iteration t such that $|\mathcal{R}| = \Lambda 2^{t-1} \geq T_{\mathsf{D\text{-}SSA}} \geq \alpha \Upsilon(\epsilon_0, \frac{\delta}{3t_{max}}) \frac{n}{\mathsf{OPT}_k}$, the inequality $2^{t-1} \geq \alpha \frac{n}{\mathsf{OPT}_k}$ will hold. The alternative will be the algorithm stops 'early' due to the condition $|\mathcal{R}_t| \geq N_{max}$ on line 16, Algorithm 3, thus, $T_2 = O(N_{max}) = O(\theta(k,t))$ and we can still conclude the efficiency of D-SSA-fix.

4 Experimental discrepancies and explanations

Huang et al. [5] shown some discrepancies in our experiments in [8]. We replicate the modifications in [5] and rerun all experiments in [8] and conclude that most anomalies found in [5] are *attributed to different experimental settings and data processing*. We were unable to reproduce some results in [5] due most likely to unknown modifications in [5].

4.1 Experimental settings

We follow the settings in [5] with the following exceptions:

- **Modifications of SSA and D-SSA.** We sent a request to the authors of [5] for the modifications they made on SSA and D-SSA, however, we have waited for 5 months without responses. Following [5], we removed all the practical optimizations we made in our code and only kept exactly what are described in our paper [8] including the larger constants, adding RIS samples one by one (not in batch of fixed size as before), starting the number of samples from Υ_1. We published our implementation in [12] for reproducibility purposes.
- **Experiment Environment.** We ran all the experiments in our Linux machine which has a 2.30Ghz Intel(R) Xeon(R) CPU E5-2650 v3 40 core processor and 256GB of RAM.
- **5 runs of each experiment:** We repeat each experiment 5 times and report the average.
- **New results of SSA and D-SSA fixes:** We also include new results for the fixes of SSA and D-SSA algorithms. The modified implementation can be found in [12].

Formating the input networks. We download most of the raw networks from the well-known Stanford SNAP dataset collection except the large network Twitter that was obtained from [7] when it was still available for download. In our original experiments in SIGMOD '16 [8], we directly took the networks and compute the edge weights according to the Weighted Cascade (WC) model. In addition, instead of using the plain-text format, we convert the network to binary format for fast I/O communication (a significant speedup, e.g. 2 minutes to read the whole Twitter network compared to almost an hour for IMM using plain-text). The performance for all comparing IM algorithms on those weighted networks are presented subsequently. Note that, we did not add the I/O time to the results on running time.

Later, we noticed that on some large undirected networks, e.g. Orkut, only one direction of the edge is stored in the raw data. However, for smaller networks, e.g. NetHEPT, NetPHY, both directions are kept. We run our experiments again and found certain matching results with [5] and suspect the discrepancies in [5] are partially caused by the data formatting. Nevertheless, all the algorithms are run on the same data set and fair comparisons are made. Note that [5] also ignored the data formatting details.

4.2 Experiments rerun

To confirm the experimental results in both our original work in SIGMOD '16 [8] and the discrepancies found in VLDB '17 [5], we replicate both of these experiments following exactly their settings as described as follows:

- **SIGMOD '16 - Rep**: Rerun of experiments in our work SIGMOD '16 [8] where the original implementations of SSA and D-SSA are used.
- **VLDB '17 -Rep**: Rerun of experiments in VLDB '17 paper [5] where we follow their descriptions to modify SSA and D-SSA algorithms, i.e. removing all the optimization we made in our original implementations.

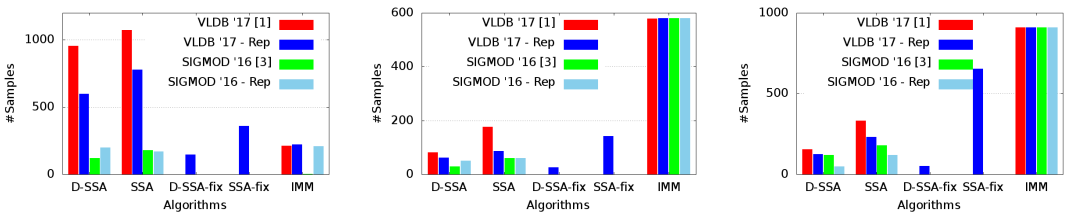
– **VLDB '17 -Rep (undirected)**: Also a rerun of experiments in VLDB '17 paper [5] but the network input is formated as undirected, i.e. for each edge (u, v), add the other direction (v, u).

For comparison, we add the results in our SIGMOD '16 paper [8] and VLDB '17 paper [5] and denote them as **SIGMOD '16 [8]** and **VLDB '17 [5]**, respectively in our results.

4.3 Possible explanations for the discrepancies

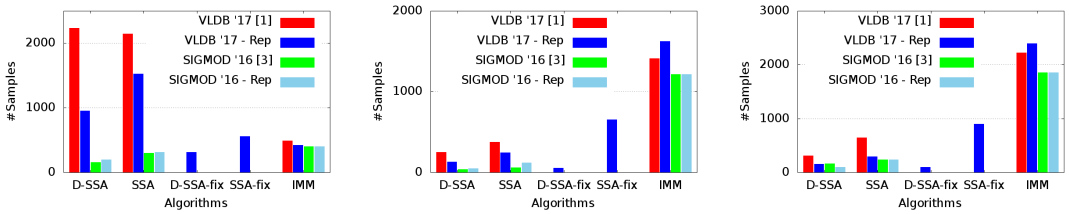
We compare the experiments' settings between our paper [8] and [5] and found the following mismatches:

– **Code modifications:** A major point is that the authors of [5] has modified our code, thus, affected the performance of our code. Since we did not received the modified code from [5] after 5 months of waiting, we follow the description in [5] to modify SSA and D-SSA.
– **Directed/Undirected network formats:** The Orkut and Friendster networks are undirected networks downloaded from SNAP library but we treat them as directed networks. All algorithms (IMM, TIM+) were run on this same directed network and, thus, the comparison were fair. The oversight of treating Orkut/Friendster as directed networks is due to the expectation that the edges in those networks are doubled, i.e., both (u, v) and (v, u) are present in the input. Unfortunately, unlike other smaller undirected networks, this is not the case for Orkut and Friendster.
– **Measures of the number of samples:** We measure the number of samples used by each algorithm. For SSA, we show the number of samples used in finding the max-coverage.



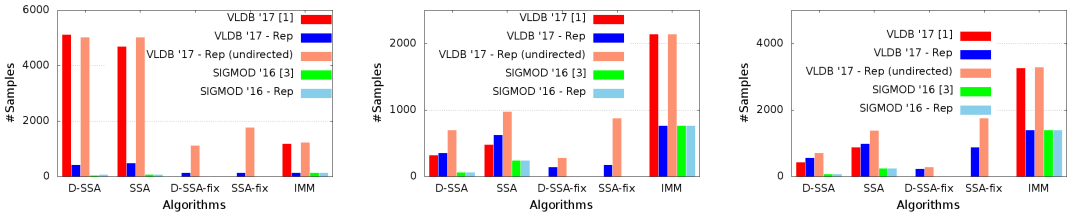
(a) $k = 1$ **(b)** $k = 500$ **(c)** $k = 1000$

Fig. 1: Number of samples generated on Enron network (See Subsection 4.2 for legend details)



(a) $k = 1$ **(b)** $k = 500$ **(c)** $k = 1000$

Fig. 2: Number of samples generated on Epinions network (See Subsection 4.2 for legend details)



(a) $k = 1$ **(b)** $k = 500$ **(c)** $k = 1000$

Fig. 3: Number of samples generated on Orkut network (See Subsection 4.2 for legend details)

4.4 Experimental results

We replicate two sets of experiments in [5] based on which they claimed the discrepancies in our experiments in [8]. The first set focuses on the number of RR sets generated by different algorithms on three networks, i.e. Enron, Epinions and Orkut, and the results are presented in Figures 1, 2 and 3. The second set of experiments is solely about the running time of IM algorithms on the case $k = 1$ and tests on 6 networks, namely NetHEPT, NetPHY, Epinions, DBLP, Orkut and Twitter. The results for these experiments are shown in Table 2. Based on our results, we draw the following observations:

• **Our results in the SIGMOD '16 paper [8] are reproducible given specific implementation settings.** The results in Figures 1, 2 and 3 show that our primary experimental results in SIGMOD '16 paper [8] are very similar to our rerun. Here we used exactly the same implementation published online in [12]. There is slight random fluctuation due to the fact that in [8], we only run each experiment once but on our rerun, we take the average over 5 runs.

• **Data processing has substantial impact on the experimental results and may cause the discrepancies found in [5].** From Figure 3 on Orkut network that we had the network format problem, we see the sharp differences when the network is formated as directed and undirected. This explains

Table 2: Relative running time of SSA, D-SSA, SSA fix and D-SSA fix to IMM for $k = 1$

Nets	LT model					IC model				
	SSA	D-SSA	SSA fix	D-SSA fix	IMM	SSA	D-SSA	SSA fix	D-SSA fix	IMM
NetHEPT	4.1	2.1	1.0	0.9	1	4.9	3.7	2.6	1.1	1
NetPHY	5.4	3.5	1.3	1.2	1	4.9	4.1	2.5	2.0	1
Epinions	3.0	2.2	1.3	1.0	1	4.3	4.0	1.2	0.9	1
DBLP	4.5	2.7	1.2	1.0	1	5.4	5.0	1.3	1.1	1
Orkut	2.1	1.4	0.7	0.7	1	5.3	4.5	1.2	1.0	1
Twitter	**0.7**	**0.6**	0.4	0.4	1	5.7	3.6	1.3	1.3	1

the discrepancies on Orkut found in the VLDB '17 paper [5]. Moreover, when formating correctly as an undirected network, our results largely agree with those in [5] on Orkut dataset.

• **Some experimental results in [5] are not replicable:** From Figures 1, 2 and 3, we see that compared to the results reported in [5], our rerun of [5] are 2 times smaller on Enron and Epinions datasets. These differences can only be explained by the unknown modifications made in [5] that were not documented in their paper and unknown to us.

On the case of $k = 1$, in [5], the authors show that IMM always runs faster than SSA and D-SSA, however, from Table 2, on the largest network, i.e. Twitter, SSA and D-SSA are faster than IMM under LT model.

The number of samples generated by SSA and D-SSA in our reruns are several times higher than that reported in our conference paper [8]. This is totally expected since we ignore all the optimizations made in our prior implementations.

• **New results: D-SSA fix, SSA fix and IMM for $k = 1$ have similar running time.** From Figures 1, 2 and 3, we also include the results for SSA fix and D-SSA fix and observe that SSA fix and D-SSA fix use fewer samples than IMM even for $k = 1$. For larger value of k, SSA fix and D-SSA fix are significantly more efficient than IMM in sample usage. From Table 2, we see that namely SSA fix and D-SSA fix use roughly the same amount of time as IMM for $k = 1$ and run faster than IMM on large networks, e.g. Orkut, Twitter, on the LT model.

References

1. Borgs, C., Brautbar, M., Chayes, J., Lucier, B.: Maximizing social influence in nearly optimal time. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. pp. 946–957. SIAM (2014)
2. Chen, W.: An issue in the martingale analysis of the influence maximization algorithm imm. arXiv preprint arXiv:1808.09363 (2018)

3. Chen, W., Lakshmanan, L.V., Castillo, C.: Information and influence propagation in social networks. Synthesis Lectures on Data Management **5**(4), 1–177 (2013)
4. Huang, K., Wang, S., Bevilacqua, G., Xiao, X., Lakshmanan, L.V.S.: Revisiting the stop-and-stare algorithms for influence maximization. `https://sites.google.com/site/vldb2017imexptr/`
5. Huang, K., Wang, S., Bevilacqua, G., Xiao, X., Lakshmanan, L.V.S.: Revisiting the stop-and-stare algorithms for influence maximization. Proceedings of the VLDB Endowment **10**(9), 913–924 (2017)
6. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 137–146. ACM (2003)
7. Kwak, H., Lee, C., Park, H.and Moon, S.: What is twitter, a social network or a news media? In: WWW. pp. 591–600. ACM (2010)
8. Nguyen, H.T., Thai, M.T., Dinh, T.N.: Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In: Proceedings of the 2016 International Conference on Management of Data. pp. 695–710. ACM (2016)
9. Nguyen, H.T., Thai, M.T., Dinh, T.N.: Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. arXiv preprint arXiv:1605.07990v2 (2016), published online 7-Sep-2016
10. Nguyen, H.T., Thai, M.T., Dinh, T.N.: A billion-scale approximation algorithm for maximizing benefit in viral marketing. IEEE/ACM Transactions on Networking (TON) **25**(4), 2419–2429 (2017)
11. Nguyen, H.T., Thai, M.T., Dinh, T.N.: Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. arXiv preprint arXiv:1605.07990 (2017), published online 22-Feb-2017
12. Nguyen, H.T., Thai, M.T., Dinh, T.N.: Ssa/dssa implementations. `https://github.com/hungnt55/Stop-and-Stare` (2018), accessed: 2018-05-16
13. Nguyen, H.T., Dinh, T.N., Thai, M.T.: Cost-aware targeted viral marketing in billion-scale networks. In: INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE. pp. 1–9. IEEE (2016)
14. Nguyen, H.T., Nguyen, T.P., Phan, N., Dinh, T.N.: Importance sketching of influence dynamics in billion-scale networks. arXiv preprint arXiv:1709.03565 (2017)
15. Tang, J., Tang, X., Yuan, J.: Influence maximization meets efficiency and effectiveness: A hop-based approach. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017. pp. 64–71. ASONAM '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3110025.3110041, `http://doi.acm.org/10.1145/3110025.3110041`
16. Tang, Y., Shi, Y., Xiao, X.: Influence maximization in near-linear time: A martingale approach. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 1539–1554. SIGMOD '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2723372.2723734, `http://doi.acm.org/10.1145/2723372.2723734`
17. Tang, Y., Xiao, X., Shi, Y.: Influence maximization: Near-optimal time complexity meets practical efficiency. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data. pp. 75–86. ACM (2014)