Outward Influence and Cascade Size Estimation in Billion-scale Networks

H. T. Nguyen, T. P. Nguyen
Virginia Commonwealth Univ.
Richmond, VA 23220
{hungnt,trinpm}@vcu.edu

T. N. Vu
Univ. of Colorado, Boulder &
UC Denver
Boulder, CO 80309
tam.vu@colorado.edu

T. N. Dinh
Virginia Commonwealth Univ.
Richmond, VA 23220
tndinh@vcu.edu

ABSTRACT

Estimating cascade size and nodes' influence is a fundamental task in social, technological, and biological networks. Yet this task is extremely challenging due to the sheer size and the structural heterogeneity of networks. We investigate a new influence measure, termed *outward influence* (OI), defined as the (expected) number of nodes that a subset of nodes S will activate, *excluding the nodes in* S. Thus, OI equals, the de facto standard measure, *influence spread* of S minus $|S|$. OI is not only more informative for nodes with small influence, but also, critical in designing new effective sampling and statistical estimation methods.

Based on OI, we propose SIEA/SOIEA, novel methods to estimate influence spread/outward influence *at scale and with rigorous theoretical guarantees*. The proposed methods are built on two novel components 1) IICP an important sampling method for outward influence; and 2) RSA, a robust mean estimation method that minimize the number of samples through analyzing variance and range of random variables. Compared to the state-of-the art for influence estimation, SIEA is $\Omega(\log^4 n)$ times faster in theory and up to *several orders of magnitude faster* in practice. For the first time, influence of nodes in the networks of billions of edges can be estimated with high accuracy within a few minutes. Our comprehensive experiments on real-world networks also give evidence against the popular practice of using a fixed number, e.g. 10K or 20K, of samples to compute the "ground truth" for influence spread.

KEYWORDS

Outward influence; FPRAS; Approximation Algorithm

ACM Reference format:
H. T. Nguyen, T. P. Nguyen, T. N. Vu, and T. N. Dinh. 2017. Outward Influence and Cascade Size Estimation in Billion-scale Networks. In *Proceedings of SIGMETRICS '17, Urbana-Champaign, IL, USA, June 05-09, 2017,* 16 pages.
DOI: 10.475/123_4

1 INTRODUCTION

In the past decade, a massive amount of data on human interactions has shed light on various cascading processes from the propagation of information and influence [17] to the outbreak of diseases [21]. These cascading processes can be modeled in graph theory through the abstraction of the network as a graph $G = (V, E)$ and a *diffusion model* that describes how the cascade proceeds into the network from a prescribed subset of nodes. A fundamental task in analyzing those cascades is to estimate the cascade size, also known as *influence spread* in social networks. This task is the foundation of the solutions for many applications including viral marketing [17, 28, 31, 32], estimating users' influence [12, 23], optimal vaccine allocation [30], identifying critical nodes in the network [11], and many others. Yet this task becomes computationally challenging in the face of the nowadays social networks that may consist of billions of nodes and edges.

S	Influence $\mathbb{I}(S)$	Outward Inf. $\mathbb{I}_{out}(S)$
$\{u\}$	$1 + p + 2p^2 = 1.12$	$p + 2p^2 = 0.12$
$\{v\}$	$1 + 2p = 1.20$	$2p = 0.20$
$\{w\}$	1.00	0.00

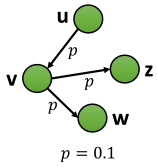


Figure 1: Left: the influence of nodes under IC model. The influence of all nodes are roughly the same, despite that w is much less influential than u and v. Right: Outward influence is better at reflecting the relative influence of the nodes. w has the least outward influence, 0, while v's is nearly twice as that of u

Most of the existing work in network cascades uses stochastic diffusion models and estimates the influence spread through sampling [8, 11, 17, 23, 29, 31]. The common practice is to use a fixed number of samples, e.g. 10K or 20K [8, 17, 29, 31], to estimate the expected size of the cascade, aka *influence spread*. Not only is there no single sample size that works well for all networks of different sizes and topologies, but those approaches also do not provide any accuracy guarantees. Recently, Lucier et al. [23] introduced INFEST, the first estimation method that comes with accuracy guarantees. Unfortunately, our experiments suggest that INFEST does not perform well in practice, taking hours on networks with only few thousand nodes. *Will there be a rigorous method to estimate the cascade size in billion-scale networks?*

In this paper, we investigate efficient estimation methods for nodes' influence under stochastic cascade models [10, 12, 17]. First, we introduce a new influence measure, called *outward influence* and defined as $\mathbb{I}_{out}(S) = \mathbb{I}(S) - |S|$, where $\mathbb{I}(S)$ denotes the influence spread. The new measure excludes the self-influence artifact in influence spread, making it *more effective in comparing relative influence of nodes*. As shown in Fig. 1, the influence spread of the nodes are roughly the same, 1. In contrast, the outward influence of nodes u, v and w are 0.12, 0.20, and 0.00, respectively. Those values correctly reflect the intuition that w is the least influential nodes and v is nearly twice as influential as u.

More importantly, the outward influence measure inspires novel methods, termed SIEA/SOIEA, to estimate influence spread/outward influence *at scale and with rigorous theoretical guarantees*. Both SOIEA and SIEA guarantee *arbitrary small relative error with high*

SIGMETRICS '17, Urbana-Champaign, IL, USA
2017. 978-1-4503-5032-7/17/06…$15.00
DOI: 10.475/123_4

probability within an $O(n)$ observed influence. The proposed methods are built on two novel components 1) IICP an important sampling method for outward influence; and 2) RSA, a robust mean estimation method that minimize the number of samples through analyzing variance and range of random variables. IICP focuses only on *non-trivial cascades* in which at least one node outside the seed set must be activated. As each IICP generates cascades of size at least two and outward influence of at least one, it leads to smaller variance and much faster convergence to the mean value. Under the well-known independent cascade model [17], SOIEA is $\Omega(\log^4 n)$ *times faster* than the state-of-the-art INFEST [23] in theory and is *four to five orders of magnitude faster* than both INFEST and the naive Monte-Carlo sampling. For other stochastic models, such as continuous-time diffusion model [12], LT model [17], SI, SIR, and variations [10], RSA can be applied directly to estimate the influence spread, given a Monte-Carlo sampling procedure, or, better, with an extension of IICP to the model.

Our contributions are summarized as follows.

- We introduce a new influence measure, called *Outward Influence* which is more effective in differentiating nodes' influence. We investigate the characteristics of this new measure including non-monotonicity, submodularity, and #P-hardness of computation.
- Two fully polynomial time randomized approximation schemes (FPRAS) SIEA and SOIEA to provide (ϵ, δ)-approximate for influence spread and outward influence with only an $O(n)$ observed influence in total. Particularly, SOIEA, our algorithm to estimate influence spread, is $\Omega(\log^4 n)$ *times faster* than the state-of-the-art INFEST [23] in theory and is *four to five orders of magnitude faster* than both INFEST and the naive Monte-Carlo sampling.
- The robust mean estimation algorithm, termed RSA, a building block of SIEA, can be used to estimate influence spread under *other stochastic diffusion models*, or, in general, mean of bounded random variables of unknown distribution. RSA will be our favorite statistical algorithm moving forwards.
- We perform comprehensive experiments on both real-world and synthesis networks with size up to 65 million nodes and *1.8 billion edges*. Our experiments indicate the superior of our algorithms in terms of both accuracy and running time in comparison to the naive Monte-Carlo and the state-of-the-art methods. The results also give *evidence against the practice of using a fixed number of samples* to estimate the cascade size. For example, using 10000 samples to estimate the influence will deviate up to 240% from the ground truth in a Twitter subnetwork. In contrast, our algorithm can provide (pseudo) *ground truth* with guaranteed small (relative) error (e.g. 0.5%). Thus it is a more concrete benchmark tool for research on network cascades.

Organization. The rest of the paper is organized as follows: In Section 2, we introduce the diffusion model and the definition of outward influence with its properties. We propose an FPRAS for outward influence estimation in Section 3. Applications in influence estimation are presented in Section 5 which is followed by the experimental results in Section 6 and conclusion in Section 8. We cover the most recent related work in Section 7.

2 DEFINITIONS AND PROPERTIES

In this section, we will introduce stochastic diffusion models, the new measure of *Outward Influence*, and showcase its properties under the popular Independent Cascade (IC) model [17].

Diffusion model. Consider a network abstracted as a graph $\mathcal{G} = (V, E)$, where V and E are the sets of nodes and edges, respectively. For example, in a social network, V and E correspond to the set of users and their social relationships, respectively. Assume that there is a cascade starting from a subset of nodes $S \subseteq V$, called *seed set*. How the cascade progress is described by a diffusion model (aka cascade model) $\mathcal{M}$ that dictates how nodes gets activated/influenced. In a stochastic diffusion model, the cascade is dictated by a random vector θ in a sample space Ω_θ. Describing the diffusion model is then equivalent to specifying the distribution P of θ.

Let $r_S(\theta)$ be the size of the cascade, the number of activated nodes in the end. The influence spread of S, denoted by $\mathbb{I}(S)$, under diffusion model $\mathcal{M}$ is the expected size of the cascade, i.e.,

$$\mathbb{I}(S) = \begin{cases} \sum_{\theta \in \Omega_\theta} r_\theta(S) \Pr[\theta] & \text{for discrete } \Omega_\theta, \\ \int_{\theta \in \Omega_\theta} r_\theta(S) dP(\theta) & \text{for continuous } \Omega_\theta \end{cases} \tag{1}$$

For example, we describe below the unknown vector θ and their distribution for the most popular diffusion models.

- Information diffusion models, e.g. Independent Cascade (IC), Linear Threshold (LT), the general triggering model [17]: $\theta \in \{0,1\}^{|E|}$, and $\forall (u,v) \in E, \theta_{(u,v)}$ is a Bernouli random variable that indicates whether u activates/influences v. That is for given $w(u,v) \in (0,1)$, $\theta(u,v) = 1$ if u activates v with a probability $w(u,v)$ and 0, otherwise.
- Epidemic cascading models, e.g., Susceptible-Infected (SI) [10, 26] and its variations: $\theta \in \mathbb{N}^{|E|}$, and $\forall (u,v) \in E, \theta_{(u,v)}$ is a random variable following a geometric distribution. $\theta_{(u,v)}$ indicates how long it takes u to activates v after u is activated.
- Continuous-time models [12]: $\theta \in \mathbb{R}^{|E|}$, and $\theta_{(u,v)}$ is a continuous random variable with density function $\pi_{u,v}(t)$. $\theta_{(u,v)}$ also indicates the transmission times (time until u activates v) like that in the SI model, however, the transmissions time on different edges follow different distributions.

Outward Influence. We introduce the notion of Outward Influence which captures the influence of a subset of nodes towards the rest of the network. Outward influence excludes the self-influence of the seed nodes from the measure.

Definition 1 (Outward Influence). *Given a graph $\mathcal{G} = (V, E)$, a set $S \subseteq V$ and a diffusion model $\mathcal{M}$, the Outward Influence of S, denoted by $\mathbb{I}_{out}(S)$, is*

$$\mathbb{I}_{out}(S) = \mathbb{I}(S) - |S| \tag{2}$$

Thus, influence and outward influence of a seed set S differ exactly by the number of nodes in S.

Influence Spread/Outward Influence Estimations. A fundemental task in network science is to estimate the influence of a given seed set S. Since the exact computation is #P-hard (Subsection 2.2), we aim for estimation with bounded error.

Definition 2 (Influence Spread Estimation). *Given a graph $\mathcal{G}$ and a set $S \subseteq V$, the problem asks for an (ϵ, δ)-estimate $\hat{\mathbb{I}}(S)$ of influence spread $\mathbb{I}(S)$, i.e.,*

$$\Pr[(1-\epsilon)\mathbb{I}(S) \leq \hat{\mathbb{I}}(S) \leq (1+\epsilon)\mathbb{I}(S)] \geq 1-\delta. \quad (3)$$

The outward influence estimation problem is stated similarly:

Definition 3 (Outward Influence Estimation). *Given a graph $\mathcal{G}$ and a set $S \subseteq V$, the problem asks for an (ϵ, δ)-estimate $\hat{\mathbb{I}}_{out}(S)$ of influence spread $\mathbb{I}_{out}(S)$, i.e.,*

$$\Pr[(1-\epsilon)\mathbb{I}_{out}(S) \leq \hat{\mathbb{I}}_{out}(S) \leq (1+\epsilon)\mathbb{I}_{out}(S)] \geq 1-\delta. \quad (4)$$

A common approach for estimation is through generating independent Monte-Carlo samples and taking the average. However, one faces two major challenges:

- How to achieve a minimum number samples to get an (ϵ, δ)-approximate?
- How to effectively generate samples with small variance, and, thus, reduce the number of samples?

For simplicity, we focus on the well-known *Independent Cascade* (IC) model and provide the extension of our approaches to other cascade models in Subsection 5.3.

2.1 Independent Cascade (IC) Model

Given a probabilistic graph $\mathcal{G} = (V, E)$ in which each edge $(u, v) \in E$ is associated with a number $w(u, v) \in (0, 1)$. $w(u, v)$ indicates the probability that node u will successfully activate v once u is activated. In practice, the probability $w(u, v)$ can be mined from interaction frequency [17, 32] or learned from action logs [13].

Cascading Process. The cascade starts from a subset of nodes $S \subseteq V$, called seed set. The cascade happens in discrete rounds $t = 0, 1, ...|V|$. At round 0, only nodes in S are active and the others are inactive. When a node u becomes active, it has a single chance to activate (aka influence) each neighbor v of u with probability $w(u, v)$. An active node remains active till the end of the cascade process. It stops when no more nodes get activated.

Sample Graph. Associate with each edge $(u, v) \in E$ a biased coin that lands heads with probability $w(u, v)$ and tails with probability $1 - w(u, v)$. Deciding the outcome when u attempts to activate v is then equivalent to the outcome of flipping the coin. If the coin landed heads, the activation attemp succeeds and we call (u, v) a *live-edge*. Since all the activation on the edges are independent in the IC model, it does not matter when we flip the coin. That is we can flip all the coins associated with the edges (u, v) at the same time instead of waiting until node u becomes active. We call the graph g that contains the nodes V and all the live-edges a *sample graph* of $\mathcal{G}$.

Note that the model parameter θ for the IC is a random vector indicating the states of the edges, i.e. *live-edge* or not. In other words, Ω_θ corresponds to the space of all possible sample graphs of $\mathcal{G}$, denoted by $\Omega_{\mathcal{G}}$.

Probabilistic Space. The graph $\mathcal{G}$ can be seen as a generative model. The set of all sample graphs generated from $\mathcal{G}$ together with their probabilities define a probabilistic space $\Omega_{\mathcal{G}}$. Recall that each sample graph $g \in \Omega_{\mathcal{G}}$ can be generated by flipping coins on all the edges to determine whether or not the edge is live or appears in g. Each edge (u, v) will be present in the a sample graph with probability $w(u, v)$. Thus, the probability that a sample graph $g = (V, E' \subseteq E)$ is generated from $\mathcal{G}$ is

$$\Pr[g \sim \mathcal{G}] = \prod_{(u,v)\in E'} w(u, v) \prod_{(u,v)\in E\setminus E'} (1 - w(u, v)). \quad (5)$$

Influence Spread and Outward Influence. In a sample graph $g \in \Omega_{\mathcal{G}}$, let $r_g(S)$ be the set of nodes reachable from S. The *influence spread* in Eq. 1 is rewritten,

$$\mathbb{I}(S) = \sum_{g\in\Omega_{\mathcal{G}}} |r_g(S)| \Pr[g \sim \mathcal{G}], \quad (6)$$

and the outward influence is defined accordingly to Eq. 2,

$$\mathbb{I}_{out}(S) = \mathbb{I}(S) - |S| \quad (7)$$

2.2 Outward Influence under IC

We show the properties of outward influence under the IC model.

Better Influence Discrepancy. As illustrated through Fig. 1, the elimination of the nominal constant $|S|$ helps to differentiate the "actual influence" of the seed nodes to the other nodes in the network. In the extreme case when $p = o(1)$, the ratio between the influence spread of u and v is $\frac{1+p+2p^2}{1+p+2p} \approx 1$, suggesting u and v have the same influence. However, outward influence can capture the fact that v can influence roughly twice the number of nodes than u, since s $\frac{\mathbb{I}_{out}(u)}{\mathbb{I}_{out}(v)} = \frac{p+2p^2}{2p} \approx 1/2$.

Non-monotonicity. Outward influence as a function of seed set S is non-monotone. This is different from the influence spread. In Figure 1, $\mathbb{I}_{out}(\{u\}) = 0.12 < \mathbb{I}_{out}(\{u, v\}) = 0.2$, however, $\mathbb{I}_{out}(\{u\}) = 0.12 > \mathbb{I}_{out}(\{u, w\}) = 0.11$. That is adding nodes to the seed set may increase or decrease the outward influence.

Submodularity. A submodular function expresses the diminishing returns behavior of set functions and are suitable for many applications, including approximation algorithms and machine learning. If Ω is a finite set, a submodular function is a set function $f : 2^\Omega \leftarrow \mathbb{R}$, where 2^Ω denotes the power set of Ω, which satisfies that for every $X, Y \subseteq \Omega$ with $X \subseteq Y$ and every $x \in \Omega \setminus Y$, we have,

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y). \quad (8)$$

Similar to influence spread, outward influence, as a function of the seed set S, is also submodular.

Lemma 1. *Given a network $G = (V, E, w)$, the outward influence function $\mathbb{I}_{out}(S)$ for $S \in 2^{|V|}$, is a submodular function*

2.3 Hardness of Computation

If we can compute outward influence of S, the influence spread of S can be obtained by adding $|S|$ to it. Since computing influence spread is #P-hard [6], it is no surprise that computing outward influence is also #P-hard.

Lemma 2. *Given a probabilistic graph $G = (V, E, w)$ and a seed set $S \subseteq V$, it is #P-hard to compute $\mathbb{I}_{out}(S)$.*

However, while influence spread is lower-bounded by one, the outward influence of any set S can be arbitrarily small (or even zero). Take an example in Figure 1, node u has influence of $\mathbb{I}(\{u\}) = 1 + p + 2p^2 \geq 1$ for any value of p. However, u's outward influence $\mathbb{I}_{out}(\{u\}) = p + 2p^2$ can be exponentially small if $p = \frac{1}{2^n}$. This makes estimating outward influence challenging, as the number of samples needed to estimate the mean of random variables is inversely proportional to the mean.

Monte-Carlo estimation. A typical approach to obtain an (ϵ, δ)-approximaion of a random variable is through Monte-Carlo estimation: taking the average over many samples of that random variable. Through the Bernstein's inequality [9], we have the lemma:

Lemma 3. *Given a set $X_1, X_2, \ldots$ of i.i.d. random variables having a common mean μ_X, there exists a Monte-Carlo estimation which gives an (ϵ, δ)-approximate of the mean μ_X and uses $T = O(\frac{1}{\epsilon^2}\ln(\frac{2}{\delta})\frac{b}{\mu_X})$ random variables where b is an upper-bound of X_i, i.e. $X_i \le b$.*

To estimate the influence spread $\mathbb{I}(S)$, existing work often simulates the cascade process using a BFS-like procedure and takes the average of the cascades' sizes as the influence spread. The number of samples needed to obtain an (ϵ, δ)-approximation is $O(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\frac{n}{\mathbb{I}(S)})$ samples. Since $\mathbb{I}(S) \ge 1$, in the worst-case, we need only a polynomial number of samples, $O(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)n)$.

Unfortunately, the same argument does not apply for the case of $\mathbb{I}_{out}(S)$, since $\mathbb{I}_{out}(S)$ can be arbitrarily close to zero. For the same reason, the recent advances in influence estimation in [3, 23] cannot be adapted to obtain a polynomial-time algorithm to compute an (ϵ, δ)-approximation (aka FPRAS) for outward influence. We shall address this challenging task in the next section.

We summarize the frequently used notations in Table 1.

Table 1: Table of notations

Notation	Description
n, m	#nodes, #edges of graph $\mathcal{G} = (V, E, w)$.
$\mathbb{I}(S)$	Influence Spread of seed set $S \subseteq V$.
$\mathbb{I}_{out}(S)$	Outward Influence of seed set $S \subseteq V$.
$N^{out}(u)$	The set of out-neighbors of u: $N^{out}(u) = \{v \in V \mid (u, v) \in E\}$
N_S^{out}	$N_S^{out} = \bigcup_{u\in S} N^{out}(u) \backslash S$.
A_i	The event that v_i is active and $v_1, \ldots, v_{i-1}$ are not active after round 1.
β_0	$\beta_0 = \sum_{i=1}^{l} \Pr[A_i] = 1 - \Pr[A_{l+1}]$.
$c(\epsilon, \delta)$	$c(\epsilon, \delta) = (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$
ϵ'	$\epsilon' = \epsilon\left(1 - \frac{\epsilon b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\right) \approx \epsilon(1 - O(\frac{1}{\ln n}))$ for $\delta = \frac{1}{n}$
Υ	$\Upsilon = (1 + \epsilon)c(\epsilon', \delta)(b - a)$

3 OUTWARD INFLUENCE ESTIMATION VIA IMPORTANCE SAMPLING

We propose a Fully Polynomial Randomized Approximation Scheme (FPRAS) to estimate the outward influence of a given set S. Given two precision parameters $\epsilon, \delta \in (0, 1)$, our FPRAS algorithm guarantees to return an (ϵ, δ)-approximate $\hat{\mathbb{I}}_{out}(S)$ of the outward influence $\mathbb{I}_{out}(S)$,

$$\Pr[(1-\epsilon)\mathbb{I}_{out}(S) \le \hat{\mathbb{I}}_{out}(S) \le (1+\epsilon)\mathbb{I}_{out}(S)] \ge 1 - \delta. \quad (9)$$

General idea. Our starting point is an observation that the cascade triggered by the seed set with small influence spread often stops right at round 0. The probability of such cascades, termed *trivial cascades*, can be computed exactly. Thus if we can sample only the *non-trivial cascades*, we will obtain a better sampling method to estimate the outward influence. The reason is that the "outward influence" associated with non-trivial cascade is also lower-bounded by one. Thus, we again can apply the argument in the previous section on the polynomial number of samples.

Given a graph $\mathcal{G}$ and a seed set S, we introduce our *importance sampling* strategy to generate such non-trivial cascades. It consists of two stages:

(1) Guarantee that at least one neighbor of S will be activated through a biased selection towards the cascades with at least one node outside of S and,
(2) Continue to simulate the cascade using the standard procedure following the diffusion model.

This importance sampling strategy is general for different diffusion models. In the following, we illustrate our importance sampling under the focused IC model.

3.1 Importance IC Polling

We propose *Importance IC Polling* (IICP) to sample non-trivial cascades in Algorithm 1.

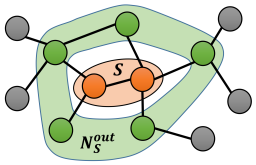


Figure 2: Neighbors of nodes in S

First, we "merge" all the nodes in S and define a "unified neighborhood" of S. Specifically, let $N^{out}(u) = \{v \mid (u, v) \in E\}$ the set of out-neighbors of u and $N_S^{out} = \bigcup_{u\in S} N_u^{out} \backslash S$ the set of out-neighbors of S *excluding* S. For each $v \in N_S^{out}$,

$$P_{S,v} = 1 - \prod_{u\in S}(1 - w(u, v)), \quad (10)$$

the probability that v is activated directly by one (or more) node(s) in S. Without loss of generality, assume that $P_{S,v} \le 1$ (otherwise, we simply add v into S).

Assume an order on the neighborhood of S, that is

$$N_S^{out} = \{v_1, v_2, \ldots, v_l\},$$

where $l = |N_S^{out}|$. For each $i = 1..l$, let A_i be the event that v_i be the "first" node that gets activated directly by S:

$$A_i = \{v_1, \ldots, v_{i-1} \text{ are not active and } v_i \text{ is active after round 1}\}.$$

The probability of A_i is

$$\Pr[A_i] = P_{S,v_i}\prod_{j=1}^{i-1}(1 - P_{S,v_j}). \quad (11)$$

For consistency, we also denote A_{l+1} the event that none of the neighbors are activated, i.e.,

$$\Pr[A_{l+1}] = 1 - \sum_{i=1}^{l}\Pr[A_i]. \quad (12)$$

Note that A_{l+1} is also the event that the cascade stops right at round 0. Such a cascade is termed a *trivial cascade*. As we can compute exactly the probability of trivial cascades, we do not need to sample those cascades but focus only on the non-trivial ones.

Denote by β_0 the probability of having at least one nodes among $v_1, \ldots, v_l$ activated by S, i.e.,

$$\beta_0 = \sum_{i=1}^{l} \Pr[A_i] = 1 - \Pr[A_{l+1}]. \quad (13)$$

Algorithm 1: IICP - Importance IC Polling

Input: A graph $\mathcal{G} = (V, E, w)$ and a seed set S
Output: $Y^{(S)}$ - size of a random outward cascade from S

Stage 1 // Sample non-trivial neighbors of set S
Precompute $\Pr[A_i]$, $i = 1, \ldots, l+1$ using Eq. 11 and Eq. 12
Select one neighbor v_i among $v_1, \ldots, v_l$ with probability of selecting v_i being $\frac{\Pr[A_i]}{\beta_0}$
Queue $R \leftarrow \{v_i\}$; $Y^{(S)} = 1$; Mark v_i and all nodes in S visited
for $j = i + 1 : l$ **do**
    With a probability P_{S,v_j} do
        Add v_j into R; $Y^{(S)} \leftarrow Y^{(S)} + 1$; Mark v_j visited.

Stage 2 // Sample from newly influenced nodes
while *R is non-empty* **do**
    $u \leftarrow R$.pop()
    foreach *unvisited neighbor* v *of* u **do**
        With a probability $w(u, v)$
            Add v to R; $Y^{(S)} \leftarrow Y^{(S)} + 1$; Mark v visited.
return $Y^{(S)}$;

We now explain the details in the Importance IC Polling Algorithm (IICP), summarized in Alg. 1. The algorithm outputs the size of the cascade minus the seed set size. We term the output of IICP the *outer size* of the cascade. The algorithm consists of two stages.

Stage 1. By definition, the events $A_i, A_2, ..., A_l, A_{l+1}$ are disjoint and form a partition of the sample space. To generate a non-trivial cascade, we first select in the first round $v_i, i = 1, \ldots, l$ with a probability $\frac{\Pr[A_i]}{\beta_0}, i = 1, \ldots, l$ (excluding A_{l+1}). This will guarantee that at least one of the neighbors of S will be activated. Let v_i be the selected node, after the first round v_i becomes active and $v_1, \ldots, v_{i-1}$ remains inactive. The nodes v_j among $v_{i+1}, \ldots, v_l$ are then activated independently with probability P_{S,v_j} (Eq. 10).

Stage 2. After the first stage of sampling neighbors of S, we get a non-trivial set of nodes directly influenced from S. For each of those nodes and later influenced nodes, we will sample a set of its neighbors by the naive BFS-like IC polling scheme [17]. Assume sampling neighbors of a newly influenced node u, each neighbor $v_j \in N^{out}(u)$ is influenced by u with probability $w(u, v_j)$. The neighbors of those influenced nodes are next to be sampled in the same fashion.

In addition, we keep track of the newly influenced nodes using a queue R and the number of active nodes outside S using $Y^{(S)}$.

The following lemma shows how to estimate the (expected) cascade size through the (expected) outer size of non-trivial cascades.

Lemma 4. *Given a seed set $S \subseteq V$, let $Y^{(S)}$ be the random variable associated with the output of the IICP algorithm. The following properties hold,*

- $1 \leq Y^{(S)} \leq n - |S|$,
- $\mathbb{I}_{out}(S) = \mathbb{E}[Y^{(S)}] \cdot \beta_0$.

Further, let Ω_W be the probability space of non-trivial cascades and Ω_Y the probability space for the outer size of non-trivial cascades, i.e, $Y^{(S)}$. The probability of $Y^{(S)} \in [1, n - |S|]$ is given by,

$$\Pr[Y^{(S)} \in \Omega_Y] = \sum_{W^{(S)} \in \Omega_W, |W^{(S)}| = Y^{(S)}} \Pr[W^{(S)} \in \Omega_W].$$

3.2 FPRAS for Outward Influence Estimation

From Lemma 4, we can obtain an estimate $\hat{\mathbb{I}}_{out}(S)$ of $\mathbb{I}_{out}(S)$ through getting an estimate $\hat{\mathbb{E}}[Y^{(S)}]$ of $\mathbb{E}[Y^{(S)}]$ by,

$$\begin{aligned}
&\Pr\left[(1-\epsilon)\mathbb{E}[Y^{(S)}] \leq \hat{\mathbb{E}}[Y^{(S)}] \leq (1+\epsilon)\mathbb{E}[Y^{(S)}]\right] \\
&= \Pr\left[(1-\epsilon)\mathbb{E}[Y^{(S)}]\beta_0 \leq \hat{\mathbb{E}}[Y^{(S)}]\beta_0 \leq (1+\epsilon)\mathbb{E}[Y^{(S)}]\beta_0\right] \\
&= \Pr\left[(1-\epsilon)\mathbb{I}_{out}(S) \leq \hat{\mathbb{I}}_{out}(S) \leq (1+\epsilon)\mathbb{I}_{out}(S)\right], \quad (14)
\end{aligned}$$

where the estimate $\hat{\mathbb{I}}_{out}(S) = \hat{\mathbb{E}}[Y^{(S)}] \cdot \beta_0$. Thus, finding an (ϵ, δ)-approximation of $\mathbb{I}_{out}(S)$ is then equivalent to finding an (ϵ, δ)-approximate $\hat{\mathbb{E}}[Y^{(S)}]$ of $\mathbb{E}[Y^{(S)}]$.

The advantage of this approach is that estimating $\mathbb{E}[Y^{(S)}]$, in which the random variable $Y^{(S)}$ has value of at least 1, requires only a polynomial number of samples. Here the same argument on the number of samples to estimate influence spread in subsection 2.3 can be applied. Let $Y_1^{(S)}, Y_2^{(S)}, \ldots$ be the random variables denoting the output of IICP. We can apply Lemma 3 on the set of random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ satisfying $1 \leq Y_i^{(S)} \leq |V| - |S|$. Since each random variable $Y_i^{(S)}$ is at least 1 and hence, $\mu_Y = \mathbb{E}[Y^{(S)}] \geq 1$, we need at most a polynomial $T = O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}(n - |S|))$ random variables for the Monte-Carlo estimation. Since, IICP has a worst-case time complexity $O(m + n)$, the Monte-Carlo using IICP is an FPRAS for estimating outward influence.

Theorem 3.1. *Given arbitrary $0 \leq \epsilon, \delta \leq 1$ and a set S, the Monte-Carlo estimation using IICP returns an (ϵ, δ)-approximation of $\mathbb{I}_{out}(S)$ using $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}(n - |S|))$ samples.*

In Section 5, we will show that both outward influence and influence spread can be estimated by a powerful algorithm saving a factor of more than $\frac{1}{\epsilon}$ random variables compared to this FPRAS estimation. The algorithm is built upon our mean estimation algorithms for bounded random variables proposed in the following.

4 EFFICIENT MEAN ESTIMATION FOR BOUNDED RANDOM VARIABLES

In this section, we propose an efficient mean estimation algorithm for bounded random variables. This is the core of our algorithms for accurately and efficiently estimating the outward influence and influence spread in Section 5.

We first propose an 'intermediate' algorithm: *Generalized Stopping Rule Estimation* (GSRA) which relies on a simple stopping rule and returns an (ϵ, δ)-approximate of the mean of lower-bounded random variables. The GSRA simultaneously generalizes and fixes the error of the Stopping Rule Algorithm [9] which only aims to estimate the mean of $[0, 1]$ random variables and has a technical error in its proof.

The main mean estimation algorithm, namely Robust Sampling Algorithm (RSA) presented in Alg. 3, effectively takes into account

both mean and variance of the random variables. It uses GSRA as a subroutine to estimate the mean value and variance at different granularity levels.

4.1 Generalized Stopping Rule Algorithm

We aim at obtaining an (ϵ, δ)-approximate of the mean of random variables $X_1, X_2, \ldots$. Specifically, the random variables are required to satisfy the following conditions:

- $a \le X_i \le b, \forall i = 1, 2, \ldots$
- $\mathbb{E}[X_{i+1}|X_1, X_2, ..., X_i] = \mu_X, \forall i = 1, 2, \ldots$

where $0 \le a < b$ are fixed constants and (unknown) μ_X.

Our algorithm generalizes the stopping rule estimation in [9] that provides (ϵ, δ) estimation of the mean of i.i.d. random variables $X_1, X_2, \ldots \in [0, 1]$. The notable differences are the following:

- We discover and amend an error in the stopping algorithm in [9]: the number of samples drawn by that algorithm may not be sufficient to guarantee the (ϵ, δ)-approximation.
- We allow estimating the mean of random variables that are *possibly dependent* and/or with *different distributions*. Our algorithm works as long as the random variables have the same means. In contrast, the algorithm in [9] can only be applied for i.i.d random variables.
- Our proposed algorithm obtains an unbiased estimator of the mean, i.e. $\mathbb{E}[\hat{\mu}_X] = \mu_X$ while [9] returns a biased one.
- Our algorithm is faster than the one in [9] whenever the lower-bound for random variables $a > 0$.

Algorithm 2: Generalized Stopping Rule Alg. (GSRA)

Input: Random variables $X_1, X_2, \ldots$ and $0 < \epsilon, \delta < 1$
Output: An (ϵ, δ)-approximate of $\mu_X = E[X_i]$

1 If $b - a < \epsilon b$, **return** $\mu_X = a$.
2 Compute: $\epsilon' = \epsilon\left(1 - \frac{\epsilon b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\right)$; $\Upsilon = (1+\epsilon)c(\epsilon', \delta)(b-a)$;
3 Initialize $h = 0$, $T = 0$;
4 **while** $h < \Upsilon$ **do**
5 $\quad h \leftarrow h + X_T$, $T \leftarrow T + 1$;
6 **return** $\hat{\mu}_X = h/T$;

Our Generalized Stopping Rule Algorithm (GSRA) is described in details in Alg. 2. Denote $c(\epsilon, \delta) = (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$.

The algorithm contains two main steps: 1) Compute the stopping threshold Υ (Line 2) which relies on the value of ϵ' computed from the given precision parameters ϵ, δ and the range $[a, b]$ of the random variables; 2) Consecutively acquire the random variables until the sum of their outcomes exceeds Υ (Line 4-5). Finally, it returns the average of the outcomes, $\hat{\mu}_X = h/T$ (Line 6), as an estimate for the mean, μ_X. Notice that Υ in GSRA depends on $(b - a)$ and thus, getting tighter bounds on the range of random variables holds a key for the efficiency of GSRA in application perspectives.

The approximation guarantee and number of necessary samples are stated in the following theorem.

Theorem 4.1. *The Generalized Stopping Rule Algorithm (GSRA) returns an (ϵ, δ)-approximate $\hat{\mu}_X$ of μ_X, i.e.,*

$$\Pr[(1-\epsilon)\mu_X \le \hat{\mu}_X \le (1+\epsilon)\mu_X] > 1 - \delta, \tag{15}$$

and, the number of samples T satisfies,

$$\Pr[T \le (1+\epsilon)\Upsilon/\mu_X] > 1 - \delta/2. \tag{16}$$

The hole in the Stopping Rule Algorithm in [9]. The estimation algorithm in [9] for estimating the mean of random variables in range $[0, 1]$ also bases on a main stopping rule condition as our GSRA. It computes a threshold

$$\Upsilon_1 = 1 + (1+\epsilon)4(e-2)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}, \tag{17}$$

where e is the base of natural logarithm, and generates samples X_j until $\sum_{j=1}^{T} X_j \ge \Upsilon_1$. The algorithm returns $\hat{\mu}_X = \frac{\Upsilon_1}{T}$ as a biased estimate of μ_X.

Unfortunately, the threshold Υ_1 to determine the stopping time does not completely account for the fact that the necessary number of samples should go over the expected one in order to provide high solution guarantees. This actually causes a flaw in their later proof of the correctness.

To amend the algorithm, we slightly strengthen the stopping condition by replacing the ϵ in the formula of Υ with an $\epsilon' = \epsilon\left(1 - \frac{\epsilon b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\right)$ (Line 2, Alg. 2). Since $\epsilon b < b - a$ (else the algorithm returns $\mu_X = a$) and assume w.l.o.g. that $\delta < 1/2$, it follows that $\epsilon' \ge 0.729\epsilon$. Thus the number of samples, in comparison to those in the stopping rule algorithm in [9] increases by at most a constant factor.

Benefit of considering the lower-bound a. By dividing the random variables by b, one can apply the stopping rule algorithm in [9] on the normalized random variables. The corresponding value of Υ is then

$$\Upsilon = 1 + (1+\epsilon)(2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2}b \tag{18}$$

Υ in our proposed algorithm is however smaller by a multiplicative factor of $\frac{b-a}{b}$. Thus it is faster than the algorithm in [9] by a factor of $\frac{b-a}{b}$ on average. Note that in case of estimating the influence, we have $a = 1, b = n - |S|$. Compared to algorithm applied [9] directly, our GSRA algorithm saves the generated samples by a factor of $\frac{b-a}{b} = \frac{n-|S|-1}{n} = 1 - \frac{|S|+1}{n} < 1$.

Martingale theory to cope with weakly-dependent random variables. To prove Theorem 4.1, we need a stronger Chernoff-like bound to deal with the general random variables $X_1, X_2, \ldots$ in range $[a, b]$ presented in the following.

Let define random variables $Y_i = \sum_{j=1}^{i}(X_j - \mu_X), \forall i \ge 1$. Hence, the random variables $Y_1, Y_2, \ldots$ form a Martingale [24] due to the following,

$$\mathbb{E}[Y_i|Y_1, \ldots, Y_{i-1}] = \mathbb{E}[Y_{i-1}] + \mathbb{E}[X_i - \mu_X] = \mathbb{E}[Y_{i-1}].$$

Then, we can apply the following lemma from [7] stating,

Lemma 5. *Let $Y_1, \ldots, Y_i, \ldots$ be a martingale, such that $|Y_1| \le \alpha$, $|Y_j - Y_{j-1}| \le \alpha$ for all $j = [2, i]$, and*

$$\mathrm{Var}[Y_1] + \sum_{j=2}^{i} \mathrm{Var}[Y_j|Y_1, \ldots, Y_{j-1}] \le \beta. \tag{19}$$

Then, for any $\lambda \ge 0$,

$$\Pr[Y_i - \mathbb{E}[Y_i] \ge \lambda] \le \exp(-\frac{\lambda^2}{2/3 \cdot \alpha \cdot \lambda + 2 \cdot \beta}) \tag{20}$$

In our case, we have $|Y_1| = |X_1 - \mu_X| \le b - a$, $|Y_j - Y_{j-1}| = |X_i - \mu_X| \le b - a$, $\mathrm{Var}[Y_1] = \mathrm{Var}[X_1 - \mu_X] = \mathrm{Var}[X]$ and $\mathrm{Var}[Y_j|Y_1, \ldots, Y_j] =$

$\mathrm{Var}[X_j - \mu_X] = \mathrm{Var}[X]$. Apply Lemma 2 with $i = T$ and $\lambda = \epsilon T \mu_X$, we have,

$$\Pr\Big[\sum_{j=1}^{T} X_j \geq (1+\epsilon)\mu_X T\Big] \leq \exp\big(\frac{-\epsilon^2 T^2 \mu_X^2}{\frac{2}{3}(b-a)\epsilon\mu_X T + 2\mathrm{Var}[X]T}\big) \tag{21}$$

Then, since $\mathrm{Var}[X] \leq \mu_X(b-\mu_X) \leq \mu_X(b-a)$ (since Bernoulli random variables with the same mean μ_X have the maximum variance), we also obtain,

$$\Pr\Big[\sum_{j=1}^{T} X_j \geq (1+\epsilon)\mu_X T\Big] \leq \exp\big(\frac{-\epsilon^2 T \mu_X}{(2+\frac{2}{3}\epsilon)(b-a)}\big). \tag{22}$$

Similarly, $-Y_1, \ldots, -Y_i, \ldots$ also form a Martingale and applying Lemma 5 gives the following probabilistic inequality,

$$\Pr\Big[\sum_{j=1}^{T} X_j \leq (1-\epsilon)\mu_X T\Big] \leq \exp\big(-\frac{\epsilon^2 T \mu_X}{2(b-a)}\big). \tag{23}$$

Algorithm 3: Robust Sampling Algorithm (RSA)

Input: Two streams of i.i.d. random variables, $X_1, X_2, \ldots$ and $X'_1, X'_2, \ldots$ and $0 < \epsilon, \delta < 1$

Output: An (ϵ, δ)-approximate $\hat{\mu}_X$ of μ_X

Step 1 // Obtain a rough estimate $\hat{\mu}'_X$ of μ_X

1 **if** $\epsilon \geq 1/4$ **then**
2 **return** $\hat{\mu}_X \leftarrow \mathrm{GSRA}(< X_1, X_2, \ldots >, \epsilon, \delta)$
3 $\hat{\mu}'_X \leftarrow \mathrm{GSRA}(< X_1, X_2, \ldots >, \sqrt{\epsilon}, \delta/3)$

Step 2 // Estimate the variance $\hat{\sigma}_X^2$

4 $\Upsilon_2 = 2\frac{1+\sqrt{\epsilon}}{1-\sqrt{\epsilon}}(1+\ln(\frac{3}{2})/\ln(\frac{2}{\delta}))\cdot\Upsilon; N_\sigma = \Upsilon_2 \cdot \epsilon/\hat{\mu}'_X; \Delta = 0;$ // Υ is defined the same as in Alg. 2
5 **for** $i = 1 : N_\sigma$ **do**
6 $\Delta \leftarrow \Delta + (X'_{2i} - X'_{2i+1})^2/2;$
7 $\hat{\rho}_X = \max\{\hat{\sigma}_X^2 = \Delta/N_\sigma, \epsilon\hat{\mu}'_X(b-a)\};$

Step 3 // Estimate μ_X

8 Set $T = \Upsilon_2 \cdot \hat{\rho}_X/(\hat{\mu}'^2_X(b-a))$, $S \leftarrow 0$;
9 **for** $i = 1 : T$ **do**
10 $S \leftarrow S + X_i$;
11 **return** $\hat{\mu}_X = S/T$;

4.2 Robust Sampling Algorithm

Our previously proposed GSRA algorithm may have problem in estimating means of random variables with small variances. An important tool that we rely on to prove the approximation guarantee in GSRA is the Chernoff-like bound in Eq. 22 and Eq. 23. However, from the inequality in Eq. 21, we can also derive the following stronger inequality,

$$\Pr\Big[\sum_{j=1}^{T} X_j \geq (1+\epsilon)\mu_X T\Big] \leq \exp\Big(\frac{-\epsilon^2 T^2 \mu_X^2}{\frac{2}{3}(b-a)\epsilon\mu_X T + 2\mathrm{Var}[X]T}\Big) \leq \exp\Big(\frac{-\epsilon^2 T \mu_X^2}{(2+\frac{2}{3})\max\{\epsilon\mu_X(b-a), \mathrm{Var}[X]\}}\Big). \tag{24}$$

In many cases, random variables have small variances and hence $\max\{\epsilon\mu_X(b-a), \mathrm{Var}[X]\} = \epsilon\mu_X(b-a)$. Thus, Eq. 24 is much stronger than Eq. 22 and can save a factor of $\frac{1}{\epsilon}$ in terms of required observed influences translating into the sample requirement. However, both the mean and variance are not available.

To achieve a robust sampling algorithm in terms of sample complexity, we adopt and improve the $\mathcal{AA}$ algorithm in [9] for general cases of $[a, b]$ random variables. The robust sampling algorithms (RSA) subsequently will estimate both the mean and variance in three steps: 1) roughly estimate the mean value with larger error ($\sqrt{\epsilon}$ or a constant); 2) use the estimated mean value to compute the number of samples necessary for estimating the variance; 3) use both the estimated mean and variance to refine the required samples to estimate mean value with desired error (ϵ, δ).

Let $X_1, X_2, \ldots$ and $X'_1, X'_2, \ldots$ are two streams of i.i.d random variables. Our robust sampling algorithm (RSA) is described in Alg. 3. It consists of three main steps:

1) If $\epsilon \geq 1/4$, run GSRA with parameter ϵ, δ and return the result (Line 1-2). Otherwise, assume $\epsilon < 1/4$ and use the Generalized Stopping Rule Algorithm (Alg. 2) to obtain an rough estimate $\hat{\mu}'_X$ using parameters of $\epsilon' = \sqrt{\epsilon} < 1/2, \delta' = \delta/3$ (Line 3).
2) Use the estimated $\hat{\mu}'_X$ in step 1 to compute the necessary number of samples, N_σ, to estimate the variance of X_i, $\hat{\sigma}_X^2$. Note that this estimation uses the second set of samples, $X'_1, X'_2, \ldots$
3) Use both $\hat{\mu}'_X$ in step 1 and $\hat{\sigma}_X^2$ in step 2 to compute the actual necessary number of samples, T, to approximate the mean μ_X. Note that this uses the same set of samples $X_1, X_2, \ldots$ as in the first step.

The numbers of samples used in the first two steps are always less than a constant times $\Upsilon \cdot \epsilon/\mu_X$ which is the minimum samples that we can achieve using the variance. This is because the first takes the error parameter $\sqrt{\epsilon}$ which is higher than ϵ and the second step uses $N_\sigma = \Upsilon_2 \cdot \epsilon/\hat{\mu}'_X$ samples.

At the end, the algorithm returns the influence estimate $\hat{\mu}_X$ which is the average over T samples, $\hat{\mu}_X = S/T$. The estimation guarantees are stated in the following theorem.

Theorem 4.2. *Let X be the probability distribution that $X_1, X_2, \ldots$ and $X'_1, X'_2, \ldots$ are drawn from. Let $\hat{\mu}_X$ be the estimate of $\mathbb{E}[X]$ returned by Alg. 3 and T be the number of drawn samples in Alg. 3 w.r.t. ϵ, δ. We have,*

(1) $\Pr[\mu_X(1-\epsilon) \leq \hat{\mu}_X \leq (1+\epsilon)\mu_X] \geq 1-\delta$,
(2) There is a universal constant c' such that

$$\Pr[T > c'\Upsilon\rho_X/(\mu_X^2(b-a))] \leq \delta \tag{25}$$

where $\rho_Z = \max\{\epsilon\mu_X(b-a), \mathrm{Var}[X]\}$.

Compared to the $\mathcal{AA}$ algorithm in [9], first of all, we replace their stopping rule algorithm with GSRA and also, we change the computation of Υ_2 which is always smaller than that of [9] by a factor of $1 + \sqrt{\epsilon} - 2\epsilon \geq 1$ when $\epsilon \leq 1/4$.

5 INFLUENCE ESTIMATION AT SCALE

This section applies our RSA algorithm to estimate both the outward influence and the traditional influence spread.

5.1 Outward Influence Estimation

We directly apply RSA algorithm on two streams of i.i.d. random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$, which are generated by IICP sampling algorithm, with the precision parameters ϵ, δ.

The algorithm is called *Scalable Outward Influence Estimation Algorithm* (SOIEA) and presented in Alg. 4 which generates two streams of random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$ (Line 1) and applies RSA algorithm on these two streams (Line 2). Note that outward influence estimate is achieved by scaling down μ_Y by β_0 (Lemma 4).

Algorithm 4: SOIEA Alg. to estimate outward influence

Input: A probabilistic graph $\mathcal{G}$, a set S and ϵ, δ
Output: $\hat{\mathbb{I}}(S)$ - an (ϵ, δ)-estimate of $\mathbb{I}(S)$
1 Generate two streams of i.i.d. random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$ by IICP algorithm.
2 **return** $\hat{\mathbb{I}}_{out}(S) \leftarrow \beta_0 \cdot \text{RSA}(< Y_1^{(S)}, \cdots >, < Y_1'^{(S)}, \cdots >, \epsilon, \delta)$

We obtain the following theoretical results incorporated from Theorem 4.2 of RSA and IICP samples.

Theorem 5.1. *The SOIEA algorithm gives an (ϵ, δ) outward influence estimation. The observed outward influences (sum of $Y^{(S)}$) and the number of generated random variables are in $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Y}{\mathbb{I}_{out}(S)/\beta_0})$ and $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Y}{\mathbb{I}^2_{out}(S)/\beta_0^2})$ respectively, where $\rho_Y = \max\{\epsilon\mathbb{I}_{out}(S)(n-|S|-1)/\beta_0, \text{Var}[Y_i^{(S)}]\}$.*

Note that $\mathbb{E}[Y^{(S)}] = \mathbb{I}_{out}(S)/\beta_0 \geq 1$.

5.2 Influence Spread Estimation

Not only is the concept of *outward influence* helpful in discriminating the relative influence of nodes but also its sampling technique, IICP, can help scale up the estimation of influence spread (IE) to billion-scale networks.

Naive approach. A naive approach is to 1) obtain an (ϵ, δ)-approximation $\hat{\mathbb{I}}_{out}(S)$ of $\mathbb{I}_{out}(S)$ using Monte-Carlo estimation 2) return $\hat{\mathbb{I}}_{out}(S) + |S|$. It is easy to show that this approach return an (ϵ, δ)-approximation for $\mathbb{I}(S)$. This approach will require $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}n)$ IICP random samples.

However, the naive approach is not optimized to estimate influence due to several reasons: 1) a loose bound $\mu_Y = \mathbb{E}[Y^{(S)}] \geq 1$ is applied to estimate outward influence; 2) casting from (ϵ, δ)-approximation of outward influence to (ϵ, δ)-approximation of influence introduces a gap that can be used to improve the estimation guarantees. We next propose more efficient algorithms based on Importance IC Sampling to achieve an (ϵ, δ)-approximate of both outward influence and influence spread. Our methods are based on two effective mean estimation algorithms.

Our approach. Based on the observations that

- $1 \leq Y^{(S)} \leq n - |S|$, i.e., we know better bounds for $Y^{(S)}$ in comparison to the cascade size which is in the range $[1, n]$.
- As we want to have an (ϵ, δ)-approximation for $Y^{(S)} + |S|$, the fixed add-on $|S|$ can be leveraged to reduce the number of samples.

We combine the effective RSA algorithm with our Importance IC Polling (IICP) for estimating the influence spread of a set S. For influence spread estimation, we will analyze random variables based on samples generated by our Importance IC Polling scheme and use those to devise an influence estimation algorithm.

Since outward influence and influence spread differ by an additive factor of $|S|$, for each outward sample $Y^{(S)}$ generated by IICP, let define a corresponding variable $Z^{(S)}$,

$$Z^{(S)} = Y^{(S)} \cdot \beta_0 + |S|, \tag{26}$$

where β_0 is defined in Eq. 13. We obtain,

- $|S| + \beta_0 \leq Z^{(S)} \leq |S| + \beta_0(n - |S|)$,
- $\mathbb{E}[Z^{(S)}] = \mathbb{E}[Y^{(S)}] \cdot \beta_0 + |S| = \mathbb{I}_{out}(S) + |S| = \mathbb{I}(S)$,

and thus we can to approximate $\mathbb{I}(S)$ by estimating $\mathbb{E}[Z^{(S)}]$.

Recall that to estimate the influence $\mathbb{I}(S)$ of a seed set S, all the previous works [6, 17, 21] resort to simulating many influence cascades from S and take the average size of those generated cascades. Let call $M^{(S)}$ the random variable representing the size of such a influence cascade. Then, we have $\mathbb{E}[M^{(S)}] = \mathbb{I}(S)$. Although both $Z^{(S)}$ and $M^{(S)}$ can be used to estimate the influence, they have different variances that lead to difference in convergence speed when estimating their means. The relation between variances of $Z^{(S)}$ and $M^{(S)}$ is stated as follows.

Lemma 6. *Let $Z^{(S)}$ defined in Eq. 26 and $M^{(S)}$ be random variable for the size of a influence cascade, the variances of $Z^{(S)}$ and $M^{(S)}$ satisfy,*

$$\text{Var}[Z^{(S)}] = \beta_0 \cdot \text{Var}[M^{(S)}] - (1 - \beta_0)\mathbb{I}^2_{out}(S) \tag{27}$$

Note that $0 \leq \beta_0 \leq 1$ and $\mathbb{I}(S) \geq |S|$. Thus, the variance of $Z^{(S)}$ is much smaller than $M^{(S)}$. Our proposed RSA on random variables X_i makes use of the variances of random variables and thus, benefits from the small variance of $Z^{(S)}$ compared to the same algorithm on the previously known random variables $M^{(S)}$.

Algorithm 5: SIEA Alg. to estimate influence spread

Input: A probabilistic graph $\mathcal{G}$, a set S and ϵ, δ
Output: $\hat{\mathbb{I}}(S)$ - an (ϵ, δ)-estimate of $\mathbb{I}(S)$
1 Generate two streams of i.i.d. random variables $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$ by IICP algorithm.
2 Compose two streams $Z_1^{(S)}, Z_2^{(S)}, \ldots$ and $Z_1'^{(S)}, Z_2'^{(S)}, \ldots$ from $Y_1^{(S)}, Y_2^{(S)}, \ldots$ and $Y_1'^{(S)}, Y_2'^{(S)}, \ldots$ using Eq. 26.
3 **return** $\hat{\mathbb{I}}(S) \leftarrow \text{RSA}(< Z_1^{(S)}, \cdots >, < Z_1'^{(S)}, \cdots >, \epsilon, \delta)$

Thus, we apply the RSA on random variables generated by IICP to develop Scalable Influence Estimation Algorithm (SIEA). SIEA is described in Alg. 5 which consists of two main steps: 1) generate i.i.d. random variables by IICP and 2) convert those variables to be used in RSA to estimate influence of S. The results are stated as follows,

Theorem 5.2. *The SIEA algorithm gives an (ϵ, δ) influence spread estimation. The observed influences (sum of random variables $Z^{(S)}$) and the number of generated random variables are in $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Z}{\mathbb{I}(S)})$ and $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Z}{\mathbb{I}^2(S)})$, where $\rho_Z = \max\{\epsilon\mathbb{I}(S)\beta_0(n-|S|-1), \text{Var}[Z_i^{(S)}]\}$.*

Comparison to INFEST [23]. Compared to the most recent state-of-the-art influence estimation in [23] that requires $O(\frac{n\log^5(n)}{\epsilon^2})$ observed influences, the SIEA algorithm incorporating IICP sampling with RSA saves at least a factor of $\log^4(n)$. That is because the necessary observed influences in SIEA is bounded by $O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\beta_0\rho_Z}{\mathbb{I}(S)})$. Since $\text{Var}[Z_i^{(S)}] \leq \mathbb{I}(S)(|S| + \beta_0(n - |S|) - \mathbb{I}(S)) \leq \mathbb{I}(S)(n - |S| - 1)$ and

hence, $\rho_Z \leq \mathbb{I}(S)(n - |S| - 1)$, when $\delta = \frac{1}{n}$ as in [23], the observed influences is then,

$$O(\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}\frac{\rho_Z}{\mathbb{I}(S)}) \leq O(\frac{n\log(2/\delta)}{\epsilon^2}) \leq O(\frac{n\log(n)}{\epsilon^2}) \quad (28)$$

Consider ϵ, δ as constants, the observed influences is $O(n)$.

5.3 Influence Spread under other Models

We can easily apply the RSA estimation algorithm to obtain an (ϵ, δ)-estimate of the influence spread under other cascade models as long as there is a Monte-Carlo sampling procedure to generate sizes of random cascades. For most stochastic diffusion models, including both discrete-time models, e.g. the popular LT with a naive sample generator described in [17], SI and SIR [10] or their variants with deadlines [26], and continuous-time models [12], designing such a Monte-Carlo sampling procedure is straightforward. Since the influence cascade sizes are at least the seed size, we always needs at most $O(n)$ samples.

To obtain more efficient sampling procedures, we can extend the idea of sampling non-trivial cascade in IICP to other models. Such sampling procedures in general will result in random variables with smaller variances and tighter bounds on the ranges. In turns, RSA, that benefits from smaller variance and range, will requires fewer samples for estimation.

5.4 Parallel Estimation Algorithms

We develop the parallel versions of our algorithms to speed up the computation and demonstrate the easy-to-parallelize property of our methods. Our main idea is that the random variable generation by IICP can be run in parallel. In particular, random variables used in each step of the core RSA algorithm can be generated simultaneously. Recall that IICP only needs to store a queue of newly active nodes, an array to mark the active nodes and a single variable $Y^{(S)}$. In total, each thread requires space in order of the number of active nodes in that simulation, $O(Y^{(S)})$, which is at most linear with size of the graph $O(n)$. In fact due to the stopping condition of linear number of observed influences, the total size of all the threads is bounded by $O(n)$ assumed the number of threads is relatively small compared to n.

Moreover, our algorithms can be implemented efficiently in terms of communication cost in distributed environments. This is because the output of IICP algorithm is just a single number $Y^{(S)}$ and thus, worker nodes in a distributed environment only communicate that single number back to the node running the estimation task. Here each IICP node holds a copy of the graph. However, the programming model needs to be considered carefully. For instance, as pointed out in many studies that the famous MapReduce is not a good fit for iterative graph processing algorithms [14, 22].

6 EXPERIMENTS

We will experimentally show that Outward Influence Estimation (SOIEA) and Outward-Based Influence Estimation (SIEA) are not only several orders of magnitudes faster than existing state-of-the-art methods but also consistently return much smaller errors. We present empirical validation of our methods on both real world and synthetic networks.

6.1 Experimental Settings

Algorithms. We compare performance of SOIEA and SIEA with the following algorithms:

- INFEST [23]: A recent influence estimation algorithm by Lucier et al. [23] in KDD'15 that provides approximation guarantees. We reimplement the algorithm in C++ accordingly to the description in [23][1].
- MC_{10K}, MC_{100K}: Variants of Monte-Carlo method that generates the traditional influence cascades [17, 21] to estimate (outward) influence spread.
- $\mathsf{MC}_{\epsilon,\delta}$: The Monte-Carlo method that uses the traditional influence cascades and guarantees (ϵ, δ)-estimation. Following [23], $\mathsf{MC}_{\epsilon,\delta}$ is only for measuring the running time of the normal Monte-Carlo to provide the same (ϵ, δ)-approximation guarantee. In particular, we obtain running time of $\mathsf{MC}_{\epsilon,\delta}$ by interpolating from that from MC_{10K}, i.e. $\frac{1}{\epsilon^2}\ln(\frac{1}{\delta})n\frac{\mathsf{Time}(\mathsf{MC}_{10K})}{10000}$.

Table 2: Datasets' Statistics

Dataset	#Nodes	#Edges	Avg. Degree
NetHEP[2]	15K	59K	4.1
NetPHY[2]	37K	181K	13.4
Epinions[2]	75K	841K	13.4
DBLP[2]	655K	2M	6.1
Orkut[2]	3M	117M	78.0
Twitter [20]	41.7M	1.5G	70.5
Friendster[2]	65.6M	1.8G	54.8

[2]From http://snap.stanford.edu

Datasets. We use both real-world networks and synthetic networks generated by GTgraph [2]. For real world networks, we choose a set of 7 datasets with sizes from tens of thousands to 65.6 millions. Table 2 gives a summary. GTgraph generates synthetic graphs with varying number of nodes and edges.

Metrics. We compare the performance of the algorithms in terms of solution quality and running time. To compare the solution quality, we adopt the relative error which shows how far the estimated number from the "ground truth". The relative error of outward influence is computed as follows:

$$|\frac{\hat{\mathbb{I}}_{out}(S)}{\mathbb{I}_{out}(S)} - 1| \cdot 100\% \quad (29)$$

where $\hat{\mathbb{I}}_{out}(S)$ is estimated outward influence of seed set S by the algorithm, $\mathbb{I}_{out}(S)$ is "ground truth" for S.

Similarly, relative error of influence spread is,

$$|\frac{\hat{\mathbb{I}}(S)}{\mathbb{I}(S)} - 1| \cdot 100\% \quad (30)$$

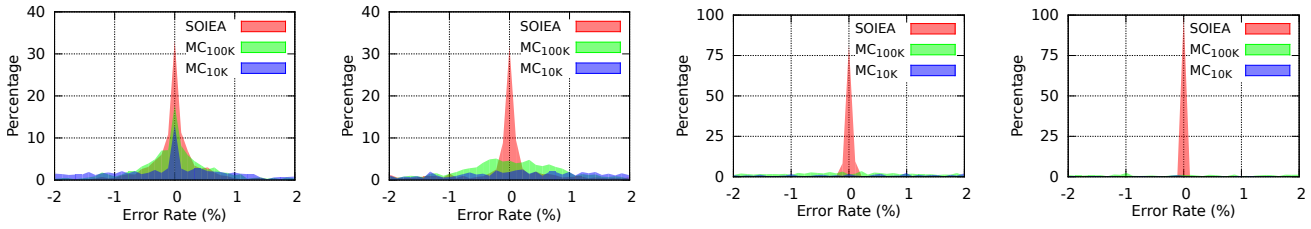
We test the algorithms on estimating different seed set sizes. For each size, we generate a set of 1000 random seed sets. We will report the average relative error (Avg. Rel. Error) and maximum relative error (Max. Rel. Error).

Ground-truth computation. We use estimates of influence and outward influence with a very small error corresponding to the setting $\epsilon = 0.005, \delta = 1/n$. We note that previous researches [23, 31] compute the "ground truth" by running Monte-Carlo with

[1]Through communication with the authors of [23], the released code has some problem and is not ready for testing.

Table 3: Comparing performance of algorithms in estimating outward influences

Dataset	Edge Models	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec)			
		SOIEA	MC_{10K}	MC_{100K}	SOIEA	MC_{10K}	MC_{100K}	SOIEA	MC_{10K}	MC_{100K}	$MC_{\epsilon,\delta}$
NetHEP	wc	0.3	1.9	0.6	2.3	25.0	8.9	0.1	0.1	0.1	12.3
	$p = 0.1$	1.0	3.7	1.2	9.7	63.0	17.2	0.2	0.1	1.0	149.5
	$p = 0.01$	0.0	4.5	1.6	0.2	20.2	9.2	0.2	0.1	0.1	8.8
	$p = 0.001$	0.0	19.2	4.6	0.1	100.0	26.4	0.2	0.1	0.1	8.5
NetPHY	wc	0.1	1.4	0.4	1.5	32.8	6.2	0.4	0.1	0.1	34.7
	$p = 0.1$	0.5	4.0	1.3	6.6	46.3	18.5	0.5	0.1	0.5	203.0
	$p = 0.01$	0.0	5.5	1.7	0.2	30.4	10.7	0.6	0.1	0.1	25.0
	$p = 0.001$	0.0	19.1	5.1	0.0	80.0	28.1	0.7	0.1	0.1	24.0



(a) WC model **(b) $p = 0.1$** **(c) $p = 0.01$** **(d) $p = 0.001$**

Figure 3: Error distributions (histogram) of the approximation errors of SOIEA, MC_{10K}, MC_{100K} on NetHEP

10,000 samples which is not sufficient as we will show later in our experiments.

Parameter Settings. For each of the datasets, we consider two common edge weighting models:

- Weighted Cascade (WC): The weight of edge (u, v) is calculated as $w(u, v) = \frac{1}{d_{in}(v)}$ where $d_{in}(v)$ denotes the in-degree of node v, as in [6, 8, 28, 31, 32].
- Constant model: All the edges has the same constant probability p as in [6, 8, 17]. We consider three different values of p, i.e. 0.1, 0.01, 0.001.

We set $\epsilon = 0.1$, $\delta = 1/n$ for SOIEA and SIEA by default or explicitly stated otherwise.

Environment. All algorithms are implemented in C++ and compiled using GCC 4.8.5. We conduct all experiments on a CentOS 7 workstation with two Intel Xeon 2.30GHz CPUs adding up to 20 physical cores and 250GB RAM.

6.2 Outward Influence Estimation

We compare SOIEA against MC_{10K} and MC_{100K} in four different edge models on NetHEP and NetPHY dataset. The results are presented in Table 3 and Figure 3.

6.2.1 Solution Quality. Table 3 illustrates that the outward influences computed by SOIEA consistently have much smaller errors in both average and maximum cases than MC_{10K} and MC_{100K} in all edge models. In particular, on NetHEP with $p = 0.001$ edge model, SOIEA has average relative error close to 0% while it is 19.2% and 4.6% for MC_{10K}, MC_{100K} respectively; the maximum relative errors of MC_{10K}, MC_{100K} in this case are 100%, 26.4% which are much higher than SOIEA of 0.1%. Apparently, MC_{100K} has smaller error rate than MC_{10K} since it uses 10 times more samples.

Figure 3 shows error distributions of SOIEA, MC_{10K}, and MC_{100K} on NetHEP. In all considered edge models, SOIEA's error highly concentrates around 0% while errors of MC_{10K} and MC_{100K} wildly spread out to a very large spectrum. In particular, SOIEA has a huge spike at the 0 error while both MC_{10K} and MC_{100K} contain two heavy tails in two sides of their error distributions. Moreover, when p gets smaller, the tails get larger as more and more empty influence simulations are generated in the traditional method.

6.2.2 Running Time. From Table 3, the running time of MC_{10K} and MC_{100K} is close to that of SOIEA while $MC_{\epsilon,\delta}$ takes up to 700 times slower than the others. Thus, in order to achieve the same approximation guarantee as SOIEA, the naive Monte-Carlo will need 700 more time than SOIEA.

Overall, SOIEA achieves significantly better solution quality and runs substantially faster than Monte-Carlo method. With larger number of samples, Monte-Carlo method can improve the quality but the running time severely suffers.

6.3 Influence Spread Estimation

This experiment evaluates SIEA by comparing its performance with the most recent state-of-the-art INFEST and naive Monte-Carlo influence estimation. Here, we use WC model to assign probabilities for the edges. We set the ϵ parameter for INFEST to 0.4 since we cannot run with smaller value of ϵ for this algorithm. Note that INFEST guarantees an error of $(1 + 8\epsilon)$, *which is equivalent to a maximum relative error of 320%.* For a fair comparison, we also run SIEA with $\epsilon = 0.4$. We use the gold-standard 10000 samples for the Monte-Carlo method (MC_{10K}). We set a time limit of 6 hours for all algorithms.

6.3.1 Solution Quality. Table 4 presents the solution quality of the algorithms in estimating size 1 seed sets, i.e. $|S| = 1$. It shows that SIEA consistently achieves substantially higher quality solution than both INFEST and MC_{10K}. Note that INFEST can only run on NetHEP and NetPHY under time limit. The average relative error of INFEST is 88 to 229 times higher than SIEA while its maximum relative error is up to 82% compared to the ground truth. The large relative error of INFEST is explained by its loose guaranteed relative error (320%). Whereas, the average relative error of MC_{10K} is up to 37 times higher than SIEA. The maximum relative error of MC_{10K}

Table 4: Comparing performance of algorithms in estimating influence spread in WC Model (seed set size $|S| = 1$)

	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec.)				
Dataset	SIEA	MC_{10K}	INFEST	SIEA	MC_{10K}	INFEST	SIEA	SIEA (16 cores)	MC_{10K}	$MC_{\epsilon,\delta}$	INFEST
NetHEP	0.2	1.2	17.7	1.5	6.6	82.7	0.1	0.1	0.0	0.8	3417.6
NetPHY	0.1	0.4	22.9	0.6	5.3	43.0	0.1	0.1	0.0	2.6	8517.7
Epinions	0.9	5.3	n/a	5.2	19.7	n/a	0.2	0.1	0.0	21.9	n/a
DBLP	0.3	1.2	n/a	1.9	8.7	n/a	2.8	1.3	0.1	770.4	n/a
Orkut	0.5	3.0	n/a	3.2	16.0	n/a	54.2	4.76	2.9	$8.2 \cdot 10^4$	n/a
Twitter	1.0	37.1	n/a	3.1	**240.8**	n/a	1272.3	106.2	7.9	$3.5 \cdot 10^6$	n/a
Friendster	0.1	3.1	n/a	0.6	23.6	n/a	1510.1	**165.1**	2.8	$2.1 \cdot 10^6$	n/a

Table 5: Comparing performance of algorithms in estimating influence spread in WC Model (seed set size $|S| = 5\%|V|$)

	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec.)				
Dataset	SIEA	MC_{10K}	INFEST	SIEA	MC_{10K}	INFEST	SIEA	SIEA (16 cores)	MC_{10K}	$MC_{\epsilon,\delta}$	INFEST
NetHEP	0.1	0.0	11.1	0.4	0.2	14.1	0.1	0.1	2.1	191.7	600.5
NetPHY	0.1	0.0	24.4	0.2	0.1	26.3	0.1	0.1	5.3	1297.1	3326.4
Epinions	0.2	0.1	20.2	0.4	0.2	23.8	0.3	0.1	20.1	$1.1 \cdot 10^4$	9325.6
DBLP	0.0	1.8	n/a	0.2	1.9	n/a	3.5	0.3	184.9	$1.0 \cdot 10^6$	n/a
Orkut	0.1	0.0	n/a	0.7	0.1	n/a	51.6	4.6	5322.8	$1.5 \cdot 10^8$	n/a
Twitter	0.2	n/a	n/a	0.5	n/a	n/a	1061.6	93.5	n/a	n/a	n/a
Friendster	0.1	n/a	n/a	0.2	n/a	n/a	2068.8	**183.1**	n/a	n/a	n/a

is up to 240% higher than the ground truth on Twitter dataset that demonstrates the insufficiency of using 10000 traditional influence samples to get the ground truth.

Differ from Table 4, Table 5 shows the results in estimating influences of seed sets of size 5% the total number of nodes. Under 6 hour limit, INFEST can only run on NetHEP, NetPHY, and Epinions while MC_{10K} could not handle the large Twitter and Friendster graph. INFEST still has a very high error compared to the other two while SIEA and MC_{10K} returns the similar quality solutions. This is because 5% of the nodes is an enormous number, i.e. > 1000000 for Friendster, and thus, the influence is huge and very few samples are needed regardless of using the traditional method or IICP.

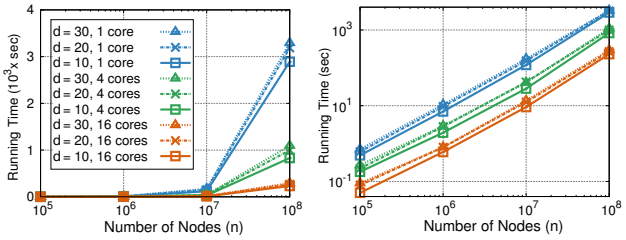
6.3.2 Running Time. In both cases of two seed set sizes, SIEA vastly outperforms $MC_{\epsilon,\delta}$ and INFEST by several orders of magnitudes. INFEST is up to 10^5 times slower than SIEA and can only run on small networks, i.e. NetHEP, NetPHY and Epinions. Compared with $MC_{\epsilon,\delta}$, the speedup factor is around 10^4, thus, MC_{10K} cannot run for the two largest networks, Twitter and Friendster in case $|S| = 5\%|V|$.

We also test the parallel version of SIEA. With 16 cores, SIEA runs about 12 times faster than that on a single core in large networks achieving an effective factor of around 75%.

Overall, SIEA consistently achieves much better solution quality and run significantly fastest than INFEST and the naive MC method. Surprisingly, under time limit of 6 hours, INFEST can only handle small networks and has very high error. The MC method achieves better accuracy for large seed sets, however, its running time increases dramatically resulting in failing to run on large datasets.

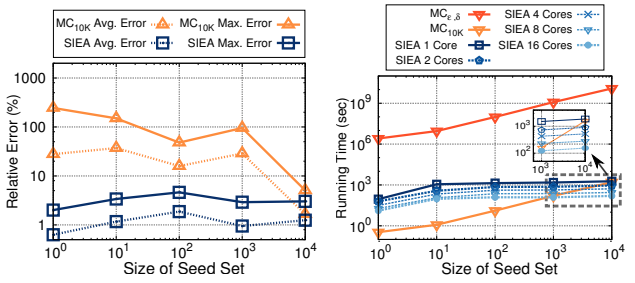
6.4 Scalability Test

We test the scalability of the single core and parallel versions of our method on synthetic networks generated by the well-known GTgraph with various network sizes. We also carry the same tests on the real-world Twitter network in comparison with the MC.



(a) Running time in linear (b) Running time in log scale

Figure 4: Running time of SIEA on synthetic networks



(a) Relative Error (b) Running Time

Figure 5: Comparing SIEA, MC_{10K} and $MC_{\epsilon,\delta}$ on Twitter

6.4.1 On Synthetic Datasets. We generate synthetic graphs using GTgraph[2], a standard graph generator used widely in large scale experiments on graph algorithms [1, 4, 15]. We generate graphs with number of nodes $n \in \{10^5, 10^6, 10^7, 10^8\}$. For each size n, we generate 3 different graphs with average degree $d \in \{10, 20, 30\}$. We use the WC model to assign edge weights. We run SIEA with different number of cores $C = \{1, 4, 16\}$

Figure 4 reports the time SIEA spent to estimate influence spread of seed set of size 1. With the same number of nodes, we see that the running time of SIEA does not significantly increase as the average degree increases. Figure 4b views Figure 4a in logarithmic scale to show the linear increase of running time with respect to the increases of nodes. As expected, SIEA speeds up proportionally

Table 6: Comparing performance of algorithms in estimating influence spread in LT model (seed set size $|S| = 1$)

	Avg. Rel. Error (%)			Max. Rel. Error (%)			Running time (sec.)				
Dataset	SIEA_{LT}	MC_{10K}	MC_{100K}	SIEA_{LT}	MC_{10K}	MC_{100K}	SIEA_{LT}	SIEA_{LT} (16 cores)	MC_{10K}	MC_{100K}	$\text{MC}_{\epsilon,\delta}$
NetHEP	1.6	1.6	0.6	8.4	7.9	2.5	0.0	0.0	0.0	0.1	1.0
NetPHY	1.2	0.5	0.3	12.7	4.4	1.4	0.0	0.0	0.0	0.1	2.9
Epinions	1.5	4.3	2.2	7.0	17.4	7.4	0.7	0.4	0.0	0.4	24.5
DBLP	0.4	1.0	0.5	5.7	11.4	2.2	2.4	0.4	0.3	2.5	1530.4
Orkut	0.5	3.3	1.1	1.9	22.1	5.9	249.4	25.0	8.5	84.2	$4.6 \cdot 10^4$
Twitter	2.4	36.1	20.7	7.1	97.5	85.6	6820.0	548.6	32.2	287.6	$1.4 \cdot 10^7$
Friendster	0.2	3.1	1.4	2.4	16.5	9.0	6183.9	701.8	20.4	137.8	$9.3 \cdot 10^6$

to number of cores used. As a result, SIEA with 16 cores is able to estimate influence spread of a random node on a synthetic graph of 100 million nodes and 1.5 billion of edges in just 5 minutes.

6.4.2 On Twitter Dataset. Figure 5 evaluates the performance of SIEA in comparison with MC_{10K} on various seed set sizes $|S| = \{1, 10, 100, 1k, 10k\}$ on Twitter dataset. On all the sizes of seed sets, SIEA consistently has average and maximum relative errors smaller than 10% (Figure 5a). The maximum relative error of MC_{10K} goes up to 244% with seed set size $|S| = 1$. As observed in experiments with large size seed sets, both SIEA and MC_{10K} have similar error rate with seed set size $|S| = 10000$.

In terms of running time, as the seed set size increases in powers of ten, SIEA's running time increases in much lower pace, e.g. few hundreds of seconds, while $\text{MC}_{\epsilon,\delta}$ consumes proportionally more time (Figure 5b). Figure 5b also evaluates parallel implementation of SIEA by varying number of CPU cores $C = \{1, 2, 4, 8, 16\}$. The running time of SIEA reduces almost two times every time the number of cores doubles confirming the almost linear speedup.

Altogether, the parallel implementation of SIEA shows a linear speedup behavior with respect to the number of cores used. On the same network with size of seed sets linearly grows, SIEA requires slightly more time to estimate influence spread while Monte-Carlo shows a linear runtime requirement. Throughout the experiments, SIEA always guarantees small error rate within ϵ.

6.5 Influence Estimation under LT Model

We illustrate the generality of our algorithms in various diffusion model by adapting SIEA for the LT model by only replacing IICP with the sampling algorithm for the LT [17]. The algorithm is then named SIEA_{LT}. The setting is similar to the case of IC. We present the results of SIEA_{LT} compared with MC_{10K}, MC_{100K}, $\text{MC}_{\epsilon,\delta}$ in Table 6. INFEST is initially proposed for the IC model, thus, we results for INFEST under the LT model are not available.

The results are mostly consistent with those observed under the IC model. SIEA_{LT} obtains significantly smaller errors and runs in order of magnitudes faster than the counterparts. The results again confirm that the estimation quality of MC using $10K$ samples is not good enough to be considered as gold-standard quality benchmark.

7 RELATED WORK

In a seminal paper [17], Kempe et al. formulated and generalized two important influence diffusion models, i.e. Independent Cascade (IC) and Linear Threshold (LT). This work has motivated a large number of follow-up researches on information diffusion [3, 6, 8, 18, 23, 29] and applications in multiple disciplines [16, 19, 21]. Kempe et al. [17] proved the monotonicity and submodularity properties of influence as a function of sets of nodes. Later, Chen et al. [6] proved that computing influence under these diffusion models is #P-hard.

Most existing works uses the naive influence cascade simulations to estimate influences [6, 17, 21, 23]. Most recently, Lucier et al. [23] proposed an estimation algorithm with rigorous quality guarantee for a single seed set. The main idea is guessing a small interval of size $(1 + \epsilon)$ that the true influence falls in and verifying whether the guess is right with high probability. However, their approach is not scalable due to a main drawback that the guessed intervals are very small, thus, the number of guesses as well as verifications made is huge. As a result, the method in [23] can only run for small dataset and still takes hours to estimate a single seed set. They also developed a distributed version on MapReduce however, graph algorithms on MapReduce have various serious issues [14, 22].

Influence estimation oracles are developed in [8, 29] which take advantage of sketching the influence to preprocess the graph for fast queries. Cohen et al. [8] use the novel bottom-k min-hash sketch to build combined reachability sketches while Ohsaka et al. in [29] adopt the reverse influence sketches. [29] also introduces the reachability-true-based technique to deal with dynamic changes in the graphs. However, these methods require days for preprocessing in order to achieve fast responses for multiple queries.

There have also been increasing interests in many related problems. [5, 13] focus on designing data mining or machine learning algorithms to extract influence cascade model parameters from real datasets, e.g. action logs. Influence Maximization, which finds a seed set of certain size with the maximum influence among those in the same size, found many real-world applications and has attracted a lot of research work [3, 6, 17, 21, 25, 27, 28, 31].

8 CONCLUSION

This paper investigates a new measure, called Outward Influence, for nodes' influence in social networks. Outward influence inspires new statiscal algorithms, namely Importance IC Polling (IICP) and Robust Mean Estimation (RSA) to estimate influence of nodes under various stochastic diffusion models. Under the popular IC model, the IICP leads to an FPRAS for estimating outward influence and SIEA to estimate influence spread. SIEA is $\Omega(\log^4(n))$ times faster than the most recent state-of-the-art and experimentally outperform the other methods by several orders of magnitudes. As previous approaches to compute ground truth influence can result in high error and long computational time, our algorithms provides concrete and scalable tools to estimate ground-truth influence for research on network cascade and social influence.

REFERENCES

[1] V. Agarwal, F. Petrini, D. Pasetto, and D. A. Bader. 2010. Scalable graph exploration on multicore processors. In *SC*. IEEE, 1–11.
[2] D. A. Bader and K. Madduri. 2006. Gtgraph: A synthetic graph generator suite. *Atlanta, GA, February* (2006).
[3] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *SODA*. SIAM, 946–957.
[4] A. Campan. 2008. A clustering approach for data and structural anonymity in social networks. *PinKDD* (2008), 54.
[5] M. Cha, A. Mislove, and K. P. Gummadi. 2009. A measurement-driven analysis of information propagation in the flickr social network. In *WWW*. ACM, 721–730.
[6] W. Chen, C. Wang, and Y. Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*. ACM, 1029–1038.
[7] F. Chung and L. Lu. 2006. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics* (2006), 79–127.
[8] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. 2014. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*. ACM, 629–638.
[9] P. Dagum, R. Karp, M. Luby, and S. Ross. 2000. An Optimal Algorithm for Monte Carlo Estimation. *SICOMP* (2000), 1484–1496.
[10] D. J. Daley, J. Gani, and J. M. Gani. 2001. *Epidemic modelling: an introduction*. Vol. 15. Cambridge University Press.
[11] T. N. Dinh and M. T. Thai. 2015. Assessing attack vulnerability in networks with uncertainty. In *INFOCOM*. IEEE, 2380–2388.
[12] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. 2013. Scalable influence estimation in continuous-time diffusion networks. In *NIPS*. 3147–3155.
[13] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. 2010. Learning Influence Probabilities in Social Networks. In *WSDM*. ACM, 241–250.
[14] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. 2013. Wtf: The who to follow service at twitter. In *WWW*. ACM, 505–514.
[15] S. Hong, Sang K. Kim, T. Oguntebi, and K. Olukotun. 2011. Accelerating CUDA graph algorithms at maximum warp. In *SIGPLAN Notices*. ACM, 267–276.
[16] Y. M. Ioannides and L. L. Datcher. 2004. Job information networks, neighborhood effects, and inequality. *Journal of economic literature* (2004), 1056–1093.
[17] D. Kempe, J. Kleinberg, and É. Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD*. 137–146.
[18] D. Kempe, J. Kleinberg, and E. Tardos. 2005. Influential nodes in a diffusion model for social networks. In *ICALP*. 1127–1138.
[19] A. Krause, A. Singh, and C. Guestrin. 2008. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR* (2008), 235–284.
[20] H. Kwak, C. Lee, H. Park, and S. Moon. 2010. What is Twitter, a social network or a news media?. In *WWW*. ACM, 591–600.
[21] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. 2007. Cost-effective outbreak detection in networks. In *KDD*. ACM, 420–429.
[22] J. Lin and M. Schatz. 2010. Design patterns for efficient graph algorithms in MapReduce. In *MLG*. ACM, 78–85.
[23] B. Lucier, J. Oren, and Y. Singer. 2015. Influence at scale: Distributed computation of complex contagion in networks. In *KDD*. ACM, 735–744.
[24] M. Mitzenmacher and E. Upfal. 2005. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press.
[25] D. T. Nguyen, Z. Huiyuan, S. Das, M. T. Thai, and T. N. Dinh. 2013. Least Cost Influence in Multiplex Social Networks: Model Representation and Analysis. In *ICDM*. 567–576.
[26] H. T. Nguyen, P. Ghosh, M. L. Mayo, and T. N. Dinh. 2016. Multiple Infection Sources Identification with Provable Guarantees. In *CIKM*. ACM, 1663–1672.
[27] H. T. Nguyen, M. T. Thai, and T. N. Dinh. 2016. Cost-aware targeted viral marketing in billion-scale networks. In *INFOCOM*. IEEE, 1–9.
[28] H. T. Nguyen, M. T. Thai, and T. N. Dinh. 2016. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In *SIGMOD*. ACM, 695–710.
[29] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. 2016. Dynamic influence analysis in evolving networks. *VLDB* (2016), 1077–1088.
[30] V. M. Preciado, M. Zargham, C. Enyioha, A. Jadbabaie, and G. Pappas. 2013. Optimal vaccine allocation to control epidemic outbreaks in arbitrary networks. In *CDC*. IEEE, 7486–7491.
[31] Y. Tang, Y. Shi, and X. Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *SIGMOD*. ACM, 1539–1554.
[32] Y. Tang, X. Xiao, and Y. Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*. ACM, 75–86.

Proof of Lemma 1

Recall that on a sampled graph $g \sim \mathcal{G}$, for a set $S \subseteq V$, we denote $r_g^{(o)}(S)$ to be the set of nodes, excluding the ones in S, that are reachable from S through live edges in g, i.e. $r_g^{(o)}(S) = r_g(S) \backslash S$. Alternatively, $r_g^{(o)}(S)$ is called the outward influence cascade of S on sample graph g and, consequently, we have,

$$\mathbb{I}_{out}(S) = \sum_{g \sim \mathcal{G}} |r_g^{(o)}(S)| \Pr[g \sim \mathcal{G}]. \tag{31}$$

It is sufficient to show that $|r_g^{(o)}(S)|$ is submodular, as $\mathbb{I}_{out}(S)$ is a linear combination of submodular functions. Consider a sample graph $g \sim \mathcal{G}$, two sets S, T such that $S \subseteq T \subseteq V$ and $v \in V \backslash T$. We have three possible cases:

- **Case** $v \in r_g^{(o)}(S)$: then $v \in r_g^{(o)}(T)$ since $S \subseteq T$ and $v \notin T$. Thus, we have the following,

$$\begin{aligned} &r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) \\ &\quad = r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T) = -1. \end{aligned} \tag{32}$$

- **Case** $v \notin r_g^{(o)}(S)$ **but** $v \in r_g^{(o)}(T)$: We have that,

$$\begin{aligned} &r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) \\ &\quad = |r_g^{(o)}(\{v\}) \backslash (r_g^{(o)}(S) \cup S)| \geq 0, \end{aligned} \tag{33}$$

while $r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T) = -1$. Thus,

$$\begin{aligned} &r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) \\ &\quad > r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T). \end{aligned} \tag{34}$$

- **Case** $v \notin r_g^{(o)}(T)$: Since $\forall u \in r_g^{(o)}(S) \cup S$, we have either $u \in r_g^{(o)}(T)$ or $u \in T$ or $r_g^{(o)}(S) \cup S \subseteq r_g^{(o)}(T) \cup T$, and thus,

$$\begin{aligned} &r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) \\ &\quad = |r_g^{(o)}(\{v\}) \backslash (r_g^{(o)}(S) \cup S)| \\ &\quad \geq |r_g^{(o)}(\{v\}) \backslash (r_g^{(o)}(T) \cup T)| \\ &\quad = r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T). \end{aligned} \tag{35}$$

In all three cases, we have,

$$\begin{aligned} &r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S) \\ &\quad \geq r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T). \end{aligned} \tag{36}$$

Applying Eq. 36 on all possible $g \sim \mathcal{G}$ and taking the sum over all of these inequalities give

$$\begin{aligned} &\sum_{g \sim \mathcal{G}} (r_g^{(o)}(S \cup \{v\}) - r_g^{(o)}(S)) \Pr[g \sim \mathcal{G}] \\ &\quad \geq \sum_{g \sim \mathcal{G}} (r_g^{(o)}(T \cup \{v\}) - r_g^{(o)}(T)) \Pr[g \sim \mathcal{G}], \end{aligned}$$

or,

$$\mathbb{I}_{out}(S \cup \{v\}) - \mathbb{I}_{out}(S) \geq \mathbb{I}_{out}(T \cup \{v\}) - \mathbb{I}_{out}(T). \tag{37}$$

That completes the proof.

Proof of Lemma 4

Let Ω_W^+ be the probability space of all possible cascades from S. For any cascade $W^{(S)} \supseteq S$, the probability of that cascade in Ω_W^+ is given by

$$\Pr[W^{(S)} \in \Omega_W^+] = \sum_{g \in \Omega_{\mathcal{G}}, g \rightsquigarrow W^{(S)}} \Pr[g \in \Omega_{\mathcal{G}}],$$

where $g \rightsquigarrow W^{(S)}$ means that $W^{(S)}$ is the set of reachable nodes from S in g.

Let Ω_W be the probability space of non-trivial cascades. According to the Stage 1 in IICP, the probability of the trivial cascade is:

$$\Pr[S \in \Omega_W] = 0.$$

Comparing to the mass of cascades in Ω_W^+, the probability mass of the trivial cascade S in Ω_W is redistributed proportionally to other cascades in Ω_W. Specifically, according to line 2 in IICP, the probability mass of all the non-trivial cascades in Ω_W is multiplied by a factor $1/\beta_0$. Thus,

$$\Pr[W^{(S)} \in \Omega_W^+] = \Pr[W^{(S)} \in \Omega_W] \cdot \beta_0 \quad \forall W^{(S)} \neq S.$$

It follows that

$$\mathbb{I}_{out}(S) = \sum_{W^{(S)} \in \Omega_W^+} |W^{(S)} \setminus S| \cdot \Pr[W^{(S)} \in \Omega_W^+] \tag{38}$$

$$= \sum_{W^{(S)} \in \Omega_W} |W^{(S)} \setminus S| \cdot \Pr[W^{(S)} \in \Omega^W] \beta_0 \tag{39}$$

$$= \mathbb{E}[|W^{(S)}|] \cdot \beta_0 = \mathbb{E}[Y^{(S)}] \cdot \beta_0. \tag{40}$$

We note that for $W^{(S)} = S$, $|W^{(S)} \setminus S| = 0$. Thus the difference in the probability masses between the two probabilistic spaces does not affect the 2nd step.

Proof of Theorem 4.1

We will equivalently prove two probabilistic inequalities:

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \frac{\delta}{2}, \tag{41}$$

and

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \frac{\delta}{2}. \tag{42}$$

Prove Eq. 41. We first realize that at termination point of Alg. 2, due to the stopping condition $h = \sum_{j=1}^{T} X_j \geq \Upsilon$ and $X_j \leq b, \forall j$, the following inequalities hold,

$$\Upsilon \leq \sum_{j=1}^{T} X_j \leq \Upsilon + b. \tag{43}$$

The left hand side of Eq. 41 is rewritten as follows,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] = \Pr\Big[\frac{\sum_{j=1}^{T} X_j}{T} < (1-\epsilon)\mu_X\Big] \tag{44}$$

$$= \Pr\Big[\sum_{j=1}^{T} X_j < (1-\epsilon)\mu_X T\Big] \tag{45}$$

$$\leq \Pr[\Upsilon < (1-\epsilon)\mu_X T]. \tag{46}$$

The last inequality is due to our realization in Eq. 43. Assume that $\epsilon < 1$ and $\mu_X > 0$, let denote $L_1 = \lceil \frac{\Upsilon}{(1-\epsilon)\mu_X} \rceil$. We then have,

$$L_1 \geq \frac{\Upsilon}{(1-\epsilon)\mu_X} \Rightarrow \frac{\Upsilon}{L_1} \leq (1-\epsilon)\mu_X, \tag{47}$$

and

$$L_1 > \frac{\Upsilon}{\mu_X} > (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2 \mu_X}(b-a). \tag{48}$$

Thus, from Eq. 46, we obtain,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \Pr[L_1 \leq T] = \Pr\Big[\sum_{j=1}^{L_1} X_j \leq \sum_{j=1}^{T} X_j\Big]$$

$$\leq \Pr\Big[\sum_{j=1}^{L_1} X_j \leq \Upsilon + b\Big] \tag{49}$$

$$\leq \Pr\Big[\frac{\sum_{j=1}^{L_1} X_j}{L_1} \leq \frac{\Upsilon + b}{L_1}\Big], \tag{50}$$

where the second inequality is due to Eq. 43. Note that $\frac{\sum_{j=1}^{L_1} X_j}{L_1}$ is an estimate of μ_X using the first L_1 random variables $X_1, \ldots, X_{L_1}$. Furthermore, from Eq. 47 that $\frac{\Upsilon}{L_1} \leq (1-\epsilon)\mu_X$, we have,

$$\frac{\Upsilon + b}{L_1} \leq (1-\epsilon)\mu_X + \frac{b}{L_1} = (1 - \epsilon + \frac{b}{L_1 \mu_X})\mu_X. \tag{51}$$

Since $L_1 > (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a)$ from Eq. 48,

$$\frac{\Upsilon + b}{L_1} \leq \Big(1 - \epsilon + \frac{\epsilon^2 b}{(2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big)\mu_X = (1-\epsilon')\mu_X. \tag{52}$$

Plugging these into Eq. 50, we obtain,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \Pr\Big[\sum_{j=1}^{L_1} X_j \leq (1-\epsilon')\mu_X L_1\Big]. \tag{53}$$

Now, apply the Chernoff-like bound in Eq. 23 with $T = L_1$ and note that $L_1 > (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a) > 2\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a)$, we achieve,

$$\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \exp\Big(-\frac{\epsilon'^2 L_1 \mu_X}{2(b-a)}\Big) \tag{54}$$

$$\leq \exp\Big(-\frac{\epsilon'^2 2\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a)}{2(b-a)}\Big)$$

$$= \frac{\delta}{2}. \tag{55}$$

That completes the proof of Eq. 41.

Prove Eq. 42. The left hand side of Eq. 42 is rewritten as follows,

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] = \Pr\Big[\sum_{j=1}^{T} X_j > (1+\epsilon)\mu_X T\Big] \tag{56}$$

$$\leq \Pr[\Upsilon + b > (1+\epsilon)\mu_X T], \tag{57}$$

where the last inequality is due to our observation that $\sum_{j=1}^{T} X_j \leq \Upsilon + b$. Under the same assumption that $0 < \mu_X \leq \frac{b}{1+\epsilon}$, we denote $L_2 = \lfloor \frac{\Upsilon + b}{(1+\epsilon)\mu_X} \rfloor$. We then have,

$$L_2 \geq \frac{\Upsilon}{(1+\epsilon)\mu_X} = (2 + \frac{2}{3}\epsilon)\ln(\frac{2}{\delta})\frac{1}{\epsilon'^2\mu_X}(b-a), \tag{58}$$

and

$$L_2 \leq \frac{\Upsilon + b}{(1+\epsilon)\mu_X} \Rightarrow \frac{\Upsilon + b}{L_2} \geq (1+\epsilon)\mu_X \tag{59}$$

$$\Rightarrow \frac{\Upsilon}{L_2} \geq (1+\epsilon)\mu_X - \frac{b}{L_2} = (1 + \epsilon - \frac{b}{L_2\mu_X})\mu_X \tag{60}$$

$$\Rightarrow \frac{\Upsilon}{L_2} \geq \Big(1 + \epsilon - \frac{\epsilon^2 b}{(2+\frac{2}{3}\epsilon)\ln(\frac{2}{\delta})(b-a)}\Big)\mu_X = (1+\epsilon')\mu_X \tag{61}$$

Thus, from Eq. 57, we obtain,

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \Pr[L_2 \geq T] = \Pr\Big[\sum_{j=1}^{L_2} X_j \geq \sum_{j=1}^{T} X_j\Big]$$

$$\leq \Pr\Big[\sum_{j=1}^{L_2} X_j \geq \Upsilon\Big] = \Pr\Big[\frac{\sum_{j=1}^{L_2} X_j}{L_2} \geq \frac{\Upsilon}{L_2}\Big] \tag{62}$$

$$\leq \Pr\Big[\frac{\sum_{j=1}^{L_2} X_j}{L_2} \geq (1+\epsilon')\mu_X\Big] \tag{63}$$

where the last inequality follows from Eq. 61. By applying another Chenoff-like bound from Eq. 22 combined with the lower bound on L_2 in Eq. 58, we achieve,

$$\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \exp\big(-\frac{\epsilon'^2 L_2 \mu_X}{(2+\frac{2}{3}\epsilon)(b-a)}\big) = \frac{\delta}{2}, \tag{64}$$

which completes the proof of Eq. 42.

Follow the same procedure as in the proof of Eq. 42, we obtain the second statement in the theorem that,

$$\Pr[T \leq (1+\epsilon)\Upsilon/\mu_X] > 1 - \delta/2, \tag{65}$$

which completes the proof of the whole theorem.

More elaboration on the hold in [9]. The stopping rule algorithm in [9] is described in Alg. 6.

Algorithm 6: Stopping Rule Algorithm [9]

Input: Random variables $X_1, X_2, \ldots$ and $0 < \epsilon, \delta < 1$
Output: An (ϵ, δ)-approximate of $\mu_X = E[X_i]$
Compute: $\Upsilon_1 = 1 + (1+\epsilon)4(e-2)\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$;
Initialize $h = 0$, $T = 0$;
while $h < \Upsilon_1$ **do**
  $h \leftarrow h + X_T$, $T \leftarrow T + 1$;
return $\hat{\mu}_X = \Upsilon_1/T$;

The algorithm first computes Υ_1 and then, generates samples X_j until the sum of their outcomes exceed Υ_1. Afterwards, it returns Υ_1/T as the estimate. Apparently, Υ_1/T is a biased estimate of μ_X since $\sum_{j=1}^{T} X_j \geq \Upsilon_1$.

An important realization for this algorithm from our proof of Theorem 4.1 is that $\Upsilon_1 \leq \sum_{j=1}^{T} X_j \leq \Upsilon_1 + b$ with $b = 1$ for $[0,1]$ random variables. In section 5 of [9], following the proof of $\Pr[\hat{\mu}_X > (1+\epsilon)\mu_X] \leq \delta/2$ to prove $\Pr[\hat{\mu}_X < (1-\epsilon)\mu_X] \leq \delta/2$, there is step that derives as follows: $\Pr[L_1 \leq T] = \Pr\Big[\sum_{j=1}^{L_1} X_j \leq \sum_{j=1}^{T} X_j\Big] = \Pr\Big[\sum_{j=1}^{L_1} X_j \leq \Upsilon_1\Big]$ where L_1 is a predefined number, i.e. $L_1 = \lfloor \frac{\Upsilon_1}{(1-\epsilon)\mu_X} \rfloor$. However, since $\Upsilon_1 \leq \sum_{j=1}^{T} X_j \leq \Upsilon_1 + b$, the last equality does not hold. This is based on Eq. 49 with the correct expression being $\Pr\Big[\sum_{j=1}^{L_1} X_j \leq \Upsilon + b\Big]$ instead of $\Pr\Big[\sum_{j=1}^{L_1} X_j \leq \Upsilon\Big]$.

Proof of Theorem 4.2

[Proof of Part (1)] If $\epsilon \geq 1/4$, then RSA only runs GSRA and hence, from Theorem 4.1, the returned solution satisfies the precision requirement. Otherwise, since the first steps is literally applying GSRA with $\sqrt{\epsilon} < 1/2, \delta/3$, we have,

$$\Pr[\mu_X(1-\sqrt{\epsilon}) \leq \hat{\mu}'_X \leq \mu_X(1+\sqrt{\epsilon})] \geq 1 - \delta/3 \tag{66}$$

We prove that in step 2, $\hat{\rho}_X \geq \rho_X/2$. Let define the random variables $\xi_i = (X'_{2i-1} - X'_{2i})^2/2, i = 1, 2, \ldots$ and thus, $\mathbb{E}[\xi_i] = \mathrm{Var}[X]$. Consider the following two cases.

(1) **If** $\mathrm{Var}[X] \geq \epsilon\mu_X(b-a)$, consider two sub-cases:
  (a) If $\mathrm{Var}[X] \geq 2(1-\sqrt{\epsilon})\epsilon\mu_X(b-a)$, then since $N_\sigma = \Upsilon_2\epsilon/\hat{\mu}'_X \geq \frac{2}{1-\sqrt{\epsilon}}(1+\ln(\frac{3}{2})/\ln(\frac{2}{\delta}))\Upsilon\epsilon/\mu_X$, applying the Chernoff-like bound in Eq. 21 gives,
  $$\Pr[\mathrm{Var}[X]/2 \leq \Delta/N_\sigma] \geq 1 - \delta/3 \tag{67}$$
  Thus, $\hat{\rho}_X \geq \mathrm{Var}[X]/2 = \rho_X/2$ with a probability of at least $1 - \delta/3$.
  (b) If $\mathrm{Var}[X] \leq 2(1-\sqrt{\epsilon})\epsilon\mu_X(b-a)$, then $\epsilon\mu_X(b-a) \geq \mathrm{Var}[X]/(2(1-\sqrt{\epsilon}))$ and therefore, $\hat{\rho}_X \geq \epsilon\hat{\mu}'_X(b-a) \geq (1-\sqrt{\epsilon})\epsilon\mu_X(b-a) \geq \mathrm{Var}_X/2 = \rho_X/2$.
(2) **If** $\mathrm{Var}[X] \leq \epsilon\mu_X(b-a)$, it follows that $\hat{\rho}_X \geq \epsilon\hat{\mu}_X \geq \rho_X(1 - \min\{\sqrt{\epsilon}, 1/2\})$ with probability at least $1 - \delta/3$.

Thus, after steps 1 and 2, $2\frac{1+\sqrt{\epsilon}}{1-\sqrt{\epsilon}}\hat{\rho}_X/\hat{\mu}'^2_X \geq \rho_X/\mu^2_Z$ with probability at least $1 - \delta/3$. In step 3, since $T = \Upsilon_2\hat{\rho}_X/(\hat{\mu}'^2_X(b-a)) \geq (1 + \ln(\frac{3}{2})/\ln(\frac{2}{\delta}))\Upsilon\rho_X/(\mu^2_X(b-a))$ and hence, applying the Chernoff-like bound in Eq. 24 again gives,

$$\Pr[\mu_X(1-\epsilon) \leq \hat{\mu}_X \leq \mu_X(1+\epsilon)] \geq 1 - 2\delta/3. \tag{68}$$

Accumulating the probabilities, we finally obtain,

$$\Pr[\mu_X(1-\epsilon) \leq \hat{\mu}_X \leq \mu_X(1+\epsilon)] \geq 1 - \delta, \tag{69}$$

This completes the proof of part (1).

[Proof of Part (2)] The RSA algorithm may fail to terminate after using $O(\Upsilon\rho_X/(\mu^2_X(b-a)))$ samples if either:

(1) The GSRA algorithm fails to return an $(\sqrt{\epsilon}, \delta/3)$-approximate $\hat{\mu}'_X$ with probability at most $\delta/2$, or,
(2) In step 2, for $\mathrm{Var}[X] \leq 2(1-\sqrt{\epsilon})\epsilon\mu_X(b-a)$, $\hat{\rho}_X$ is not $O(\epsilon\mu_X(b-a))$ with probability at most $\delta/2$.

From Theorem 4.1, with $T = (1+\epsilon)\Upsilon/\mu_X = O(\Upsilon\rho_X/(\mu^2_X(b-a)))$, the first case happens with probability at most $\delta/2$. In addition, we can show similarly to Theorem 4.1 that if $\mathrm{Var}[X] \leq 2\epsilon\mu_X(b-a)$, then,

$$\Pr[\Delta/T \geq 4\epsilon\mu_X(b-a)] \leq \exp(-T\epsilon\mu_X(b-a)/2). \tag{70}$$

Thus, for $T \geq 2\Upsilon\epsilon/\mu_X$, we have $\Pr[\Delta/T \geq 4\epsilon\mu_X] \leq \delta/2$.

Proof of Lemma 6

We start with the computation of $\mathrm{Var}[Z^{(S)}]$ with a note that $\mathbb{E}[Z^{(S)}] = \mathbb{I}(S)$,

$$\begin{aligned}
\mathrm{Var}[Z^{(S)}] &= \sum_{z=\beta_0+|S|}^{(n-|S|)\beta_0+|S|} (z - \mathbb{E}[Z^{(S)}])^2 \Pr[Z^{(S)} = z] \\
&= \sum_{y=1}^{n-|S|} (y\beta_0 + |S| - \mathbb{I}(S))^2 \Pr[Y^{(S)} = y] \\
&= \sum_{y=1}^{n-|S|} (y\beta_0 - \mathbb{I}_{out}(S)\beta_0 + \mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))^2 \Pr[Y^{(S)} = y] \\
&= \beta_0^2 \sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S))^2 \Pr[Y^{(S)} = y] \\
&\quad + \sum_{y=1}^{n-|S|} (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))^2 \Pr[Y^{(S)} = y] \\
&\quad + 2\beta_0 \sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S))(\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S)) \Pr[Y^{(S)} = y]
\end{aligned}$$

Since $Y^{(S)} \geq 1$ and $\Pr[Y^{(S)} = y] = \frac{\Pr[M^{(S)} = y+|S|]}{\beta_0}$, we have,

$$\begin{aligned}
&\sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S))^2 \Pr[Y^{(S)} = y] \\
&\quad = \frac{1}{\beta_0} \sum_{m=1+|S|}^{n} (m - \mathbb{E}[M^{(S)}])^2 \Pr[M^{(S)} = m] \\
&\quad = \frac{1}{\beta_0} \sum_{m=|S|}^{n} (m - \mathbb{E}[M^{(S)}])^2 \Pr[M^{(S)} = m] - \frac{1}{\beta_0}\mathbb{I}_{out}^2(S)(1-\beta_0) \\
&\quad = \frac{1}{\beta_0}(\mathrm{Var}[M^{(S)}] - \mathbb{I}_{out}^2(S)(1-\beta_0)), \qquad (71)
\end{aligned}$$

and,

$$\begin{aligned}
&\sum_{y=1}^{n-|S|} \beta_0 (y - \mathbb{I}_{out}(S))(\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S)) \Pr[Y^{(S)} = y] \\
&\quad = (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S)) \sum_{y=1}^{n-|S|} (y - \mathbb{I}_{out}(S)) \Pr[Y^{(S)} = y] \\
&\quad = (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))\mathbb{I}_{out}(S)(1/\beta_0 - 1). \qquad (72)
\end{aligned}$$

Plug these back in the $\mathrm{Var}[Z^{(S)}]$, we obtain,

$$\begin{aligned}
\mathrm{Var}[Z^{(S)}] &= \beta_0(\mathrm{Var}[M^{(S)}] - \mathbb{I}_{out}^2(S)(1-\beta_0)) \\
&\quad + (\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))^2 \\
&\quad + 2\beta_0(\mathbb{I}_{out}(S)\beta_0 + |S| - \mathbb{I}(S))\mathbb{I}_{out}(S)(1/\beta_0 - 1) \\
&= \beta_0 \cdot \mathrm{Var}[M^{(S)}] - (1-\beta_0)\mathbb{I}_{out}^2(S) \qquad (73)
\end{aligned}$$

That completes the computation.