Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency

Youze Tang Xiaokui Xiao Yanchen Shi
School of Computer Engineering
Nanyang Technological University
Singapore
tangyouze@gmail.com xkxiao@ntu.edu.sg shiy0017@e.ntu.edu.sg

ABSTRACT

Given a social network G and a constant k, the *influence maximization* problem asks for k nodes in G that (directly and indirectly) influence the largest number of nodes under a pre-defined diffusion model. This problem finds important applications in viral marketing, and has been extensively studied in the literature. Existing algorithms for influence maximization, however, either trade approximation guarantees for practical efficiency, or vice versa. In particular, among the algorithms that achieve constant factor approximations under the prominent *independent cascade (IC)* model or *linear threshold (LT)* model, none can handle a million-node graph without incurring prohibitive overheads.

This paper presents *TIM*, an algorithm that aims to bridge the theory and practice in influence maximization. On the theory side, we show that *TIM* runs in $O((k+\ell)(n+m)\log n/\varepsilon^2)$ expected time and returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1-n^{-\ell}$ probability. The time complexity of *TIM* is near-optimal under the IC model, as it is only a $\log n$ factor larger than the $\Omega(m+n)$ lower-bound established in previous work (for fixed k, ℓ, and ε). Moreover, *TIM* supports the *triggering model*, which is a general diffusion model that includes both IC and LT as special cases. On the practice side, *TIM* incorporates novel heuristics that significantly improve its empirical efficiency without compromising its asymptotic performance. We experimentally evaluate *TIM* with the largest datasets ever tested in the literature, and show that it outperforms the state-of-the-art solutions (with approximation guarantees) by up to four orders of magnitude in terms of running time. In particular, when $k=50$, $\varepsilon=0.2$, and $\ell=1$, *TIM* requires less than one hour on a commodity machine to process a network with 41.6 million nodes and 1.4 billion edges. This demonstrates that influence maximization algorithms can be made practical while still offering strong theoretical guarantees.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms, Theory, Experimentation

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

ACM 978-1-4503-2376-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2588555.2593670.

1. INTRODUCTION

Let G be a social network, and M be a probabilistic model that captures how the nodes in G may *influence* each other's behavior. Given G, M, and a small constant k, the *influence maximization* problem asks for the k nodes in G that can (directly and indirectly) influence the largest number of nodes. This problem finds important applications in *viral marketing* [8,25], where a company selects a few *influential* individuals in a social network and provides them with incentives (e.g., free samples) to adopt a new product, hoping that the product will be recursively recommended by each individual to his/her friends to create a large cascade of further adoptions.

Kempe et al. [17] are the first to formulate influence maximization as a combinatorial optimization problem. They consider several probabilistic cascade models from the sociology and marketing literature [13–15, 27], and present a general greedy approach that yields $(1-1/e-\varepsilon)$-approximate solutions for all models considered, where ε is a constant. This seminal work has motivated a large body of research on influence maximization in the past decade [2–7, 10, 16–19, 21, 28, 30, 31].

Kempe et al.'s greedy approach is well accepted for its simplicity and effectiveness, but it is known to be computationally expensive. In particular, it has an $\Omega\left(kmn\cdot\mathrm{poly}\left(\varepsilon^{-1}\right)\right)$ time complexity [3] where n and m are the numbers of nodes and edges in the social network, respectively. Empirically, it runs in days even when n and m are merely a few thousands [6]. Such inefficiency of Kempe et al.'s method has led to a plethora of algorithms [5–7,10,16,19,21,30,31] that aim to reduce the computation overhead of influence maximization. Those algorithms, however, either trade performance guarantees for practical efficiency, or vice versa. In particular, most algorithms rely on heuristics to efficiently identify nodes with large influence, but they fail to achieve any approximation ratio under Kempe et al.'s cascade models; there are a few exceptions [6,11,21] that retain the $(1-1/e-\varepsilon)$-approximation guarantee, but they have the same time complexity with Kempe et al.'s method and still cannot handle large networks.

Very recently, Borgs et al. [3] make a theoretical breakthrough and present an $O(k\ell^2(m+n)\log^2 n/\varepsilon^3)$ time algorithm[1] for influence maximization under the *independent cascade (IC)* model, i.e., one of the prominent models from Kempe et al. [17]. Borgs et al. show that their algorithm returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1-n^{-\ell}$ probability, and prove that it is *near-optimal* since any other algorithm that provides the same approximation guarantee and succeeds with at least a constant probability must run in $\Omega(m+n)$ time [3]. Although Borgs et al.'s algorithm significantly improves upon previous methods in terms of asymptotic performance, its practical efficiency is rather unsatisfactory, due to a large hidden constant factor in its time complexity. In short, no existing influence maximization algorithm can scale

to million-node graphs while still providing non-trivial approximation guarantees (under Kempe et al.'s models [17]). Therefore, any practitioner who conducts influence maximization on sizable social networks can only resort to heuristics, even though the results thus obtained could be arbitrarily worse than the optimal ones.

Our Contributions. This paper presents *Two-phase Influence Maximization (TIM)*, an algorithm that aims to bridge the theory and practice in influence maximization. On the theory side, we show that *TIM* returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1-n^{-\ell}$ probability, and it runs in $O((k+\ell)(m+n)\log n/\varepsilon^2)$ expected time. The time complexity of *TIM* is near-optimal under the IC model, as it is only a $\log n$ factor larger than the $\Omega(m+n)$ lower-bound established by Borgs et al. [3] (for fixed k, ℓ, and ε). Moreover, *TIM* supports the *triggering model* [17], which is a general cascade model that includes the IC model as a special case.

On the practice side, *TIM* incorporates novel heuristics that result in up to 100-fold improvements of its computation efficiency, without any compromise of theoretical assurances. We experimentally evaluate *TIM* with a variety of social networks, and show that it outperforms the state-of-the-art solutions (with approximation guarantees) by up to four orders of magnitude in terms of running time. In particular, when $k = 50$, $\varepsilon \geq 0.2$, and $\ell = 1$, *TIM* requires less than one hour to process a network with 41.6 million nodes and 1.4 billion edges. To our knowledge, this is the first result in the literature that demonstrates efficient influence maximization on a billion-edge graph.

In summary, our contributions are as follows:

1. We propose an influence maximization algorithm that runs in near-linear expected time and returns $(1 - 1/e - \varepsilon)$-approximate solutions (with a high probability) under the triggering model.
2. We devise several optimization techniques that improve the empirical performance of our algorithm by up to 100-fold.
3. We provide theoretical analysis on the state-of-the-art solutions with approximation guarantees, and establish the superiority of our algorithm in terms of asymptotic performance.
4. We experiment with the largest datasets ever used in the literature, and show that our algorithm can efficiently handle graphs with more than a billion edges. This demonstrates that influence maximization algorithms can be made practical while still offering strong theoretical guarantees.

2. PRELIMINARIES

In this section, we formally define the influence maximization problem, and present an overview of Kempe et al. and Borgs et al.'s solutions [3, 17]. For ease of exposition, we focus on the *independent cascade (IC)* model [17] considered by Borgs et al. [3]. In Section 4.2, we discuss how our solution can be extended to the more general *triggering model*.

2.1 Problem Definition

Let G be a social network with a node set V and a directed edge set E, with $|V| = n$ and $|E| = m$. Assume that each directed edge e in G is associated with a *propagation probability* $p(e) \in [0, 1]$. Given G, the independent cascade (IC) model considers a time-stamped influence propagation process as follows:

[1]The time complexity of Borgs et al.'s algorithm is established as $O(\ell^2(m+n)\log^2 n/\varepsilon^3)$ in [3], but our correspondence with Borg et al. shows that it should be revised as $O(k\ell^2(m+n)\log^2 n/\varepsilon^3)$, due to a gap in the proof of Lemma 3.6 in [3].

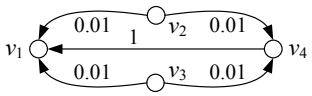


Figure 1: Social network G.

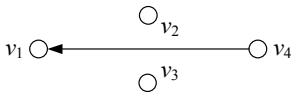


Figure 2: Random graph g_1.

1. At timestamp 1, we *activate* a selected set S of nodes in G, while setting all other nodes *inactive*.
2. If a node u is first activated at timestamp i, then for each directed edge e that points from u to an inactive node v (i.e., v is an inactive outgoing neighbor of u), u has $p(e)$ probability to activate v at timestamp $i + 1$. After timestamp $i + 1$, u cannot activate any node.
3. Once a node becomes activated, it remains activated in all subsequent timestamps.

Let $I(S)$ be the number of nodes that are activated when the above process converges, i.e., when no more nodes can be activated. We refer to S as the *seed set*, and $I(S)$ as the *spread* of S. Intuitively, the influence propagation process under the IC model mimics the spread of an infectious disease: the seed set S is conceptually similar to an initial set of infected individuals, while the activation of a node by its neighbors is analogous to the transmission of the disease from one individual to another.

For example, consider a propagation process on the social network G in Figure 1, with $S = \{v_2\}$ as the seed set. (The number on each edge indicates the propagation probability of the edge.) At timestamp 1, we activate v_2, since it is only node in S. Then, at timestamp 2, both v_1 and v_4 have 0.01 probability to be activated by v_2, as (i) they are both v_2's outgoing neighbors and (ii) the edges from v_2 to v_1 and v_4 have a propagation probability of 0.01. Suppose that v_2 activates v_4 but not v_1. After that, at timestamp 3, v_4 will activate v_1 since the edge from v_4 to v_1 has a propagation probability of 1. After that, the influence propagation process terminates, since no other node can be activated. The total number of nodes activated during the process is 3, and hence, $I(S) = 3$.

Given G and a constant k, the influence maximization problem under the IC model asks for a size-k seed set S with the maximum expected spread $\mathbb{E}[I(S)]$. In other words, we seek a seed set that can (directly and indirectly) activate the largest number of nodes in expectation.

2.2 Kempe et al.'s Greedy Approach

In a nutshell, Kempe et al.'s approach [17] (referred to as *Greedy* in the following) starts from an empty seed set $S = \emptyset$, and then iteratively adds into S the node u that leads to the largest increase in $\mathbb{E}[I(S)]$, until $|S| = k$. That is,

$$u \quad = \quad \arg\max_{v \in V} \Big(\mathbb{E}\big[I\big(S \cup \{v\}\big)\big] - \mathbb{E}\big[I(S)\big]\Big).$$

Greedy is conceptually simple, but it is non-trivial to implement since the computation of $\mathbb{E}[I(S)]$ is #P-hard [5]. To address this issue, Kempe et al. propose to estimate $\mathbb{E}[I(S)]$ to a reasonable accuracy using a Monte Carlo method. To explain, suppose that we flip a coin for each edge e in G, and remove the edge with $1 - p(e)$ probability. Let g be the resulting graph, and $R(S)$ be the set of nodes in g that are *reachable* from S. (We say that a node v in g is reachable from S, if there exists a directed path in g that starts from a node in S and ends at v.) Kempe et al. prove that the expected size of $R(S)$ equals $\mathbb{E}[I(S)]$. Therefore, to estimate $\mathbb{E}[I(S)]$, we can first generate multiple instances of g, then measure $R(S)$ on each instance, and finally take the average measurement as an estimation of $\mathbb{E}[I(S)]$.

Assume that we take a large number r of measurements in the estimation of each $\mathbb{E}[I(S)]$. Then, with a high probability, *Greedy* yields a $(1 - 1/e - \varepsilon)$-approximate solution under the IC model [17], where ε is a constant that depends on both G and r [3, 18]. In general, *Greedy* achieves the same approximation ratio under any cascade model where $\mathbb{E}[I(S)]$ is a *submodular* function of S [18]. To our knowledge, however, there is no formal analysis in the literature on how r should be set to achieve a given ε on G. Instead, Kempe et al. suggest setting $r = 10000$, and most follow-up work adopts similar choices of r. In Section 5, we provide a formal result on the relationship between ε and r.

Although *Greedy* is general and effective, it incurs significant computation overheads due to its $O(kmnr)$ time complexity. Specifically, it runs in k iterations, each of which requires estimating the expected spread of $O(n)$ node sets. In addition, each estimation of expected spread takes measurements on r graphs, and each measurement needs $O(m)$ time. These lead to an $O(kmnr)$ total running time.

2.3 Borgs et al.'s Method

The main reason for *Greedy*'s inefficiency is that it requires estimating the expected spread of $O(kn)$ node sets. Intuitively, most of those $O(kn)$ estimations are *wasted* since, in each iteration of *Greedy*, we are only interested in the node set with the largest expected spread. Yet, such wastes of computation are difficult to avoid under the framework of *Greedy*. To explain, consider the first iteration of *Greedy*, where we are to identify a single node in G with the maximum expected spread. Without prior knowledge on the expected spread of each node, we would have to evaluate $\mathbb{E}[I(\{v\})]$ for each node v in G. In that case, the overhead of the first iteration alone would be $O(mnr)$.

Borgs et al. [3] avoid the limitation of *Greedy* and propose a drastically different method for influence maximization under the IC model. We refer to the method as *Reverse Influence Sampling (RIS)*. To explain how *RIS* works, we first introduce two concepts:

DEFINITION 1 (REVERSE REACHABLE SET). *Let v be a node in G, and g be a graph obtained by removing each edge e in G with $1 - p(e)$ probability. The reverse reachable (RR) set for v in g is the set of nodes in g that can reach v. (That is, for each node u in the RR set, there is a directed path from u to v in g.)*

DEFINITION 2 (RANDOM RR SET). *Let $\mathcal{G}$ be the distribution of g induced by the randomness in edge removals from G. A* random RR set *is an RR set generated on an instance of g randomly sampled from $\mathcal{G}$, for a node selected uniformly at random from g.*

By definition, if a node u appears in an RR set generated for a node v, then u can reach v via a certain path in G. As such, u should have a chance to activate v if we run an influence propagation process on G using $\{u\}$ as the seed set. Borgs et al. show a result that is consistent with the above observation: If an RR set generated for v has ρ probability to overlap with a node set S, then when we use S as the seed set to run an influence propagation process on G, we have ρ probability to activate v (See Lemma 2). Based on this result, Borgs et al.'s *RIS* algorithm runs in two steps

1. Generate a certain number of random RR sets from G.
2. Consider the *maximum coverage* problem [29] of selecting k nodes to cover the maximum number of RR sets generated[2]. Use the standard greedy algorithm [29] to derive a $(1-1/e)$-approximate solution S_k^* for the problem. Return S_k^* as the final result.

[2]We say that a node v covers a set of nodes S if and only if $v \in S$.

The rationale of *RIS* is as follows: If a node u appears in a large number of RR sets, then it should have a high probability to activate many nodes under the IC model; in that case, u's expected spread should be large. By the same reasoning, if a size-k node set S_k^* covers most RR sets, then S_k^* is likely to have the maximum expected spread among all size-k node sets in G. In that case, S_k^* should be a good solution to influence maximization. We illustrate *RIS* with an example.

EXAMPLE 1. Consider that we invoke *RIS* on the social network G in Figure 1, setting $k = 1$. *RIS* first generates a number of random RR sets, each of which is pertinent to (i) a node sampled uniformly at random from G and (ii) a random graph obtained by removing each edge e in G with $1 - p(e)$ probability (see Definition 2). Assume that the first RR set R_1 is pertinent to v_1 and the random graph g_1 in Figure 2. Then, we have $R_1 = \{v_1, v_4\}$, since v_1 and v_4 are the only two nodes in g_1 that can reach v_1.

Suppose that, besides R_1, *RIS* only constructs three other random RR sets R_2, R_3, and R_4, which are pertinent to three random graphs g_2, g_3, and g_4, respectively. For simplicity, assume that (i) g_2, g_3, and g_4 are identical to g_1, and (ii) the node that *RIS* samples from g_i ($i \in [2, 4]$) is v_i. Then, we have $R_2 = \{v_2\}$, $R_3 = \{v_3\}$, and $R_4 = \{v_4\}$. In that case, v_4 is the node that covers the most number of RR sets, since it appears in two RR sets (i.e., R_1 and R_4), whereas any other node only covers one RR set. Consequently, *RIS* returns $S_k^* = \{v_4\}$ as the result. □

Compared with *Greedy*, *RIS* can be more efficient as it avoids estimating the expected spreads of a large number of node sets. That said, we need to carefully control the number of random RR sets generated in Step 1 of *RIS*, so as to strike a balance between efficiency and accuracy. Towards this end, Borgs et al. propose a threshold-based approach: they allow *RIS* to keep generating RR sets, until the total number of nodes and edges examined during the generation process reaches a pre-defined threshold τ. They show that when τ is set to $\Theta(k(m + n) \log n/\varepsilon^3)$, *RIS* runs in time linear to τ, and it returns a $(1 - 1/e - \varepsilon)$-approximate solution to the influence maximization problem with at least a constant probability. They then provide an algorithm that amplifies the success probability to at least $1 - n^{-\ell}$, by increasing τ by a factor of ℓ, and repeating *RIS* for $\Omega(\ell \log n)$ times.

Despite of its near-linear time complexity, *RIS* still incurs significant computational overheads in practice, as we show in Section 7. The reason can be intuitively explained as follows. Given that *RIS* sets a threshold τ on the total cost of Step 1, the RR sets sampled in Step 1 are correlated, due to which some nodes in G may appear in RR sets more frequently than normal[3]. In that case, even if we identify a node set that covers the most number of RR sets, it still may not be a good solution to the influence maximization problem. Borgs et al. mitigate the effects of correlations by setting τ to a large number. However, this not only results in the ε^{-3} term in *RIS*'s time complexity, but also renders *RIS*'s practical efficiency less than satisfactory.

3. PROPOSED SOLUTION

This section presents *TIM*, an influence maximization method that borrows ideas from *RIS* but overcomes its limitations with a

[3]To demonstrate this phenomenon, imagine that we repeatedly sample from a Bernoulli distribution with $p = 0.5$, until the sum of samples reaches 1. It can be verified that our sample set has $1/2$ probability to contain more 1 than 0, but only $1/4$ probability to contain more 0 than 1. In other words, 1 is the most frequent number in the sample set with an abnormally high probability.

Notation	Description
G, G^T	a social network G, and its *transpose* G^T constructed by exchanging the starting and ending points of each edge in G
n	the number of nodes in G (resp. G^T)
m	the number of edges in G (resp. G^T)
k	the size of the seed set for influence maximization
$p(e)$	the propagation probability of an edge e
$I(S)$	the spread of a node set S in an influence propagation process on G (see Section 2.2)
$w(R)$	the number of edges in G^T that starts from the nodes in an RR set R (see Equation 1)
$\kappa(R)$	see Equation 8
$\mathcal{R}$	the set of all RR sets generated in Algorithm 1
$F_{\mathcal{R}}(S)$	the fraction of RR sets in $\mathcal{R}$ that are covered by a node set S
EPT	the expected width of a random RR set
OPT	the maximum $I(S)$ for any size-k node set S
KPT	a lower-bound of OPT established in Section 3.2
λ	see Equation 4

Table 1: Frequently used notations.

novel algorithm design. At a high level, *TIM* consists of two phases as follows:

1. **Parameter Estimation.** This phase computes a lower-bound of the maximum expected spread among all size-k node sets, and then uses the lower-bound to derive a parameter θ.

2. **Node Selection.** This phase samples θ random RR sets from G, and then derives a size-k node set S_k^* that covers a large number of RR sets. After that, it returns S_k^* as the final result.

The node selection phase of *TIM* is similar to *RIS*, except that it samples a *pre-decided* number (i.e., θ) of random RR sets, instead of using a threshold on computation cost to indirectly control the number. This ensures that the RR sets generated by *TIM* are independent (given θ), thus avoiding the correlation issue that plagues *RIS*. Meanwhile, the derivation of θ in the parameter estimation phase is non-trivial: As we shown in Section 3.1, θ needs to be larger than a certain threshold to ensure the correctness of *TIM*, but the threshold depends on the optimal result of influence maximization, which is unknown. To address this challenge, we compute a θ that is above the threshold but still small enough to ensure the overall efficiency of *TIM*.

In what follows, we first elaborate the node selection phase of *TIM*, and then detail the parameter estimation phase. For ease of reference, Table 1 lists the notations frequently used. Unless otherwise specified, all logarithms in this paper are to the base e.

3.1 Node Selection

Algorithm 1 presents the pseudo-code of *TIM*'s node selection phase. Given G, k, and a constant θ, the algorithm first generates θ random RR sets, and inserts them into a set $\mathcal{R}$ (Lines 1-2). The subsequent part of the algorithm consists of k iterations (Lines 3-7). In each iteration, the algorithm selects a node v_j that covers the largest number of RR sets in $\mathcal{R}$, and then removes all those covered RR sets from $\mathcal{R}$. The k selected nodes are put into a set S_k^*, which is returned as the final result.

Implementation. Lines 6-10 in Algorithm 1 correspond to a standard greedy approach for a *maximum coverage* problem [29], i.e., the problem of selecting k nodes to cover the largest number of node sets. It is known that this greedy approach returns $(1-1/e)$-approximate solutions, and has a linear-time implementation. For brevity, we omit the description of the implementation and refer interested readers to [3] for details.

Algorithm 1 NodeSelection (G, k, θ)

1: Initialize a set $\mathcal{R} = \emptyset$.
2: Generate θ random RR sets and insert them into $\mathcal{R}$.
3: Initialize a node set $S_k^* = \emptyset$.
4: **for** $j = 1$ to k **do**
5: Identify the node v_j that covers the most RR sets in $\mathcal{R}$.
6: Add v_j into S_k^*.
7: Remove from $\mathcal{R}$ all RR sets that are covered by v_j.
8: **return** S_k^*

Meanwhile, the generation of each RR set in Algorithm 1 is implemented as a randomized breath-first search (BFS) on G. Given a node v in G, we first create an empty queue, and then flip a coin for each *incoming* edge e of v; with $p(e)$ probability, we retrieve the node u from which e starts, and we put u into the queue. Subsequently, we iteratively extract the node v' at the top of the queue, and examine each incoming edge e' of v; if e' starts from an unvisited node u', we add u' into the queue with $p(e')$ probability. This iterative process terminates when the queue becomes empty. Finally, we collect all nodes visited during the process (including v), and use them to form an RR set.

Performance Bounds. We define the *width* of an RR set R, denoted as $w(R)$, as the number of directed edges in G whose point to the nodes in R. That is

$$w(R) = \sum_{v \in R} (\text{the indegree of } v \text{ in } G). \tag{1}$$

Observe that if an edge is examined in the generation of R, then it must point to a node in R. Let EPT be the expected width of a random RR set. It can be verified that Algorithm 1 runs in $O(\theta \cdot EPT)$ time. In the following, we analyze how θ should be set to minimize the expected running time while ensuring solution quality. Our analysis frequently uses the Chernoff bounds [24]:

LEMMA 1. *Let X be the sum of c i.i.d. random variables sampled from a distribution on $[0,1]$ with a mean μ. For any $\delta > 0$,*

$$Pr\Big[X - c\mu \geq \delta \cdot c\mu\Big] \leq \exp\left(-\frac{\delta^2}{2+\delta} c\mu\right),$$

$$Pr\Big[X - c\mu \leq -\delta \cdot c\mu\Big] \leq \exp\left(-\frac{\delta^2}{2} c\mu\right).$$

In addition, we utilize the following lemma from [3] that establishes the connection between RR sets and the influence propagation process on G:

LEMMA 2. *Let S be a fixed set of nodes, and v be a fixed node. Suppose that we generate an RR set R for v on a graph g that is constructed from G by removing each edge e with $1 - p(e)$ probability. Let ρ_1 be the probability that S overlaps with R, and ρ_2 be the probability that S, when used as a seed set, can activate v in an influence propagation process on G. Then, $\rho_1 = \rho_2$.*

The proofs of all theorems, lemmas, and corollaries in Section 3 are included in the appendix.

Let $\mathcal{R}$ be the set of all RR sets generated in Algorithm 1. For any node set S, let $F_{\mathcal{R}}(S)$ be fraction of RR sets in $\mathcal{R}$ covered by S. Then, based on Lemma 2, we can prove that the expected value of $n \cdot F_{\mathcal{R}}(S)$ equals the expected spread of S in G:

COROLLARY 1. $\mathbb{E}[n \cdot F_{\mathcal{R}}(S)] = \mathbb{E}[I(S)]$.

Let OPT be the maximum expected spread of any size-k node set in G. Using the Chernoff bounds, we show that $n \cdot F_{\mathcal{R}}(S)$ is an accurate estimator of any node set S's expected spread, when θ is sufficiently large:

LEMMA 3. *Suppose that θ satisfies*

$$\theta \geq (8+2\varepsilon)n \cdot \frac{\ell \log n + \log \binom{n}{k} + \log 2}{OPT \cdot \varepsilon^2}. \quad (2)$$

Then, for any set S of at most k nodes, the following inequality holds with at least $1 - n^{-\ell}/\binom{n}{k}$ probability:

$$\left| n \cdot F_{\mathcal{R}}(S) - \mathbb{E}[I(S)] \right| < \frac{\varepsilon}{2} \cdot OPT. \quad (3)$$

Based on Lemma 3, we prove that when Equation 2 holds, Algorithm 1 returns a $(1 - 1/e - \varepsilon)$-approximate solution with high probability:

THEOREM 1. *Given a θ that satisfies Equation 2, Algorithm 1 returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1 - n^{-\ell}$ probability.*

Notice that it is difficult to set θ directly based on Equation 2, since OPT is unknown. We address this issue in Section 3.2, by presenting an algorithm that returns a θ which not only satisfies Equation 2, but also leads to an $O((k+\ell)(m+n)\log n/\varepsilon^2)$ expected time complexity for Algorithm 1. For simplicity, we define

$$\lambda = (8+2\varepsilon)n \cdot \left(\ell \log n + \log \binom{n}{k} + \log 2\right) \cdot \varepsilon^{-2}, \quad (4)$$

and we rewrite Equation 2 as

$$\theta \geq \lambda / OPT. \quad (5)$$

3.2 Parameter Estimation

Recall that the expected time complexity of Algorithm 1 is $O(\theta \cdot EPT)$, where EPT is the expected number of coin tosses required to generate an RR set for a randomly selected node in G. Our objective is to identify an θ that makes $\theta \cdot EPT$ reasonably small, while still ensuring $\theta \geq \lambda/OPT$. Towards this end, we first define a probability distribution $\mathcal{V}^*$ over the nodes in G, such that the probability mass for each node is proportional to its in-degree in G. Let v^* be a random variable following $\mathcal{V}^*$. We have the following lemma:

LEMMA 4. $\frac{n}{m} EPT = \mathbb{E}[I(\{v^*\})]$, *where the expectation of $I(\{v^*\})$ is taken over the randomness in v^* and the influence propagation process.*

In other words, if we randomly sample a node from $\mathcal{V}^*$ and calculate its expected spread s, then on average we have $s = \frac{n}{m} EPT$. This implies that $\frac{n}{m} EPT \leq OPT$, since OPT equals the maximum expected spread of any size-k node set.

Suppose that we are able to identify a number t such that $t = \Omega(\frac{n}{m} EPT)$ and $t \leq OPT$. Then, by setting $\theta = \lambda/t$, we can guarantee that Algorithm 1 is correct and has an expected time complexity of

$$O(\theta \cdot EPT) = O\left(\frac{m}{n}\lambda\right) = O\left((k+\ell)(m+n)\log n/\varepsilon^2\right). \quad (6)$$

Choices of t. An intuitive choice of t is $t = \frac{n}{m} EPT$, since (i) both n and m are known and (ii) EPT can be estimated by measuring the average width of RR sets. However, we observe that when $k \gg 1$, $t = \frac{n}{m} EPT$ renders $\theta = \lambda/t$ unnecessarily large, which in turn leads to inferior efficiency. To explain, recall that $\frac{n}{m} EPT$ equals the mean of the expected spread of a node v^* sampled from $\mathcal{V}^*$, and hence, it is independent of k. In contrast, OPT increases monotonically with k. Therefore, the difference between $\frac{n}{m} EPT$ and OPT increases with k, which makes $t = \frac{n}{m} EPT$ an unfavorable choice of t when k is large. To tackle this problem, we replace $\frac{n}{m} EPT$ with a closer approximation of OPT that increases with k, as explained in the following.

Algorithm 2 KptEstimation (G, k)

```
1: for i = 1 to log_2 n − 1 do
2:     Let c_i = (6ℓ log n + 6 log(log_2 n)) · 2^i.
3:     Let sum = 0.
4:     for j = 1 to c_i do
5:         Generate a random RR set R.
6:         κ(R) = 1 − (1 − w(R)/m)^k
7:         sum = sum + κ(R).
8:     if sum/c_i > 1/2^i then
9:         return KPT* = n · sum/(2 · c_i)
10: return KPT* = 1
```

Suppose that we take k samples from $\mathcal{V}^*$, and use them to form a node set S^*. (Note that S^* may contain fewer than k nodes due to the elimination of duplicate samples.) Let KPT be the mean of the expected spread of S^* (over the randomness in S^* and the influence propagation process). It can be verified that

$$\frac{n}{m} EPT \leq KPT \leq OPT, \quad (7)$$

and that KPT increases with k. We also have the following lemma:

LEMMA 5. *Let R be a random RR set and $w(R)$ be the width of R. Define*

$$\kappa(R) = 1 - \left(1 - \frac{w(R)}{m}\right)^k. \quad (8)$$

Then, $KPT = n \cdot \mathbb{E}[\kappa(R)]$, where the expectation is taken over the random choices of R.

By Lemma 5, we can estimate KPT by first measuring $n \cdot \kappa(R)$ on a set of random RR sets, and then taking the average of the measurements. But how many measurements should we taken? By the Chernoff bounds, if we are to obtain an estimate of KPT with $\delta \in (0,1)$ relative error with at least $1 - n^{-\ell}$ probability, then the number of measurements should be $\Omega(\ell n \log n \cdot \delta^{-2}/KPT)$. In other words, the number of measurements required depends on KPT, whereas KPT is exactly the subject being measured. We resolve this dilemma with an adaptive sampling approach that dynamically adjusts the number of measurements based on the observed samples of RR sets.

Estimation of KPT. Algorithm 2 presents our sampling approach for estimating KPT. The high level idea of the algorithm is as follows. We first generate a relatively small number of RR sets, and use them to derive an estimation of KPT with a bounded absolute error. If the estimated value of KPT is much larger than the error bound, we infer that the estimation is accurate enough, and we terminate the algorithm. On the other hand, if the estimated value of KPT is not large compared with the error bound, then we generate more RR sets to obtain a new estimation of KPT with a reduced absolute error. After that, we re-evaluate the accuracy of our estimation, and if necessary, we further increase the number of RR sets, until a precise estimation of KPT is computed.

More specifically, Algorithm 2 runs in at most $\log_2 n - 1$ iterations. In the i-th iteration, it samples c_i RR sets from G (Lines

2-7), where

$$c_i = \left(6\ell \log n + 6 \log \left(\log_2 n\right)\right) \cdot 2^i. \quad (9)$$

Then, it measures $\kappa(R)$ on each RR set R, and computes the average value of $\kappa(R)$. Our choice of c_i ensures that if this average value is larger than 2^{-i}, then with a high probability, $\mathbb{E}[\kappa(R)]$ is at least half of the average value; in that case, the algorithm terminates by returning a KPT^* that equals the average value times $n/2$ (Lines 8-9). Meanwhile, if the average value is no more than 2^{-i}, then the algorithm proceeds to the $(i+1)$-th iteration.

On the other hand, if the average value is smaller than 2^{-i} in all $\log_2 n - 1$ iterations, then the algorithm returns $KPT^* = 1$, which equals the smallest possible KPT (since each node in the seed set can always activate itself). As we show shortly, $\mathbb{E}[\frac{1}{KPT^*}] = O(\frac{1}{KPT})$, and $KPT^* \in [KPT/4, OPT]$ holds with a high probability. Hence, setting $\theta = \lambda / KPT^*$ ensures that Algorithm 1 is correct and achieves the expected time complexity in Equation 6.

Theoretical Analysis. Although Algorithm 2 is conceptually simple, proving its correctness and effectiveness is non-trivial as it requires a careful analysis of the algorithm's behavior in each iteration. In what follows, we present a few supporting lemmas, and then use them to establish Algorithm 2's performance guarantees.

Let $\mathcal{K}$ be the distribution of $\kappa(R)$ over random RR sets in G. Then, $\mathcal{K}$ has a domain $[0, 1]$. Let $\mu = KPT/n$, and s_i be the sum of c_i i.i.d. samples from $\mathcal{K}$, where c_i is as defined in Equation 9. By the Chernoff bounds, we have the following result:

LEMMA 6. *If $\mu \leq 2^{-j}$, then for any $i \in [1, j-1]$,*

$$\Pr\left[\frac{s_i}{c_i} > \frac{1}{2^i}\right] < \frac{1}{n^\ell \cdot \log_2 n}.$$

By Lemma 6, if $KPT \leq 2^{-j}$, then Algorithm 2 is very unlikely to terminate in any of the first $j-1$ iterations. This prevents the algorithm from outputting a KPT^* too much larger than KPT.

LEMMA 7. *If $\mu \geq 2^{-j}$, then for any $i \geq j+1$,*

$$\Pr\left[\frac{s_i}{c_i} > \frac{1}{2^i}\right] > 1 - n^{-\ell \cdot 2^{i-j-1}} / \log_2 n.$$

By Lemma 7, if $KPT \leq 2^{-j}$ and Algorithm 2 happens to enter its $i > j+1$ iteration, then it will almost surely terminate in the i-th iteration. This ensures that the algorithm would not output a KPT^* that is considerably smaller than KPT.

Based on Lemmas 6 and 7, we prove the following theorem on the accuracy and expected time complexity of Algorithm 2:

THEOREM 2. *When $n \geq 2$ and $\ell \geq 1/2$, Algorithm 2 returns $KPT^* \in [KPT/4, OPT]$ with at least $1 - n^{-\ell}$ probability, and runs in $O(\ell(m+n)\log n)$ expected time. Furthermore, $\mathbb{E}\left[\frac{1}{KPT^*}\right] < \frac{12}{KPT}$.*

3.3 Putting It Together

In summary, our *TIM* algorithm works as follows. Given G, k, and two parameters ε and ℓ, *TIM* first feeds G and k as input to Algorithm 2, and obtains a number KPT^* in return. After that, *TIM* computes $\theta = \lambda / KPT^*$, where λ is as defined in Equation 4 and is a function of k, ℓ, n, and ε. Finally, *TIM* gives G, k, and θ as input to Algorithm 1, whose output S_k^* is the final result of influence maximization.

By Theorems 1 and 2, Equation 6, and the union bound, *TIM* runs in $O((k+\ell)(m+n)\log n/\varepsilon^2)$ expected time, and returns a $(1 - 1/e - \varepsilon)$-approximate solution with at least $1 - 2 \cdot n^{-\ell}$ probability. This success probability can easily increased to $1 - n^{-\ell}$, by scaling ℓ up by a factor of $1 + \log 2 / \log n$. Finally, we note that the time complexity of *TIM* is near-optimal under the IC model, as it is only a $\log n$ factor larger than the $\Omega(m+n)$ lower-bound proved by Borgs et al. [3] (for fixed k, ℓ, and ε).

Algorithm 3 RefineKPT (G, k, KPT^*, ε')

1: Let $\mathcal{R}'$ be the set of all RR sets generated in the last iteration of Algorithm 2.
2: Initialize a node set $S_k' = \emptyset$.
3: **for** $j = 1$ to k **do**
4: Identify the node v_j that covers the most RR sets in $\mathcal{R}'$.
5: Add v_j into S_k'.
6: Remove from $\mathcal{R}'$ all RR sets that are covered by v_j.
7: Let $\lambda' = (2+\varepsilon')\ell n \log n \cdot (\varepsilon')^{-2}$.
8: Let $\theta' = \lambda' / KPT^*$.
9: Generate θ' random RR sets; put them into a set $\mathcal{R}''$.
10: Let f be the fraction of the RR sets in $\mathcal{R}''$ that is covered by S_k'.
11: Let $KPT' = f \cdot n / (1+\varepsilon')$
12: **return** $KPT^+ = \max\{KPT', KPT^*\}$

4. EXTENSIONS

In this section, we present a heuristic method for improving the practical performance of *TIM* (without affecting its asymptotic guarantees), and extend *TIM* to an influence propagation model more general than the IC model.

4.1 Improved Parameter Estimation

The efficiency of *TIM* highly depends on the output KPT^* of Algorithm 2. If KPT^* is close to OPT, then $\theta = \lambda / KPT^*$ is small; in that case, Algorithm 1 only needs to generate a relatively small number of RR sets, thus reducing computation overheads. However, we observe that KPT^* is often much smaller than OPT on real datasets, which severely degrades the efficiency of Algorithm 1 and the overall performance of *TIM*.

Our solution to the above problem is to add an intermediate step between Algorithms 1 and 2 to refine KPT^* into a (potentially) much tighter lower-bound of OPT. Algorithm 3 shows the pseudo-code of the intermediate step. The algorithm first retrieves the set $\mathcal{R}'$ of all RR sets created in the last iteration of Algorithm 2, i.e., the RR sets that from which KPT^* is computed. Then, it invokes the greedy approach (for the maximum coverage problem) on $\mathcal{R}'$, and obtains a size-k node set S_k' that covers a large number of RR sets in $\mathcal{R}'$ (Lines 2-6 in Algorithm 3).

Intuitively, S_k' should have a large expected spread, and thus, if we can estimate $\mathbb{E}[I(S_k')]$ to a reasonable accuracy, then we may use the estimation to derive a good lower-bound for OPT. Towards this end, Algorithm 3 generates a number θ' of random RR sets, and examine the fraction f of RR sets that are covered by S_k' (Lines 7-10). By Corollary 1, $f \cdot n$ is an unbiased estimation of $\mathbb{E}[I(S_k')]$. We set θ' to a reasonably large number to ensure that $f \cdot n < (1+\varepsilon') \cdot \mathbb{E}[I(S_k')]$ occurs with at most $1 - n^{-\ell}$ probability. Based on this, Algorithm 3 computes $KPT' = f \cdot n / (1+\varepsilon')$, which scales $f \cdot n$ down by a factor of $1 + \varepsilon'$ to ensure that $KPT' \leq \mathbb{E}[I(S_k')] \leq OPT$. The final output of Algorithm 3 is $KPT^+ = \max\{KPT', KPT^*\}$, i.e., we choose the larger one between KPT' and KPT^* as the new lower-bound for OPT. The following lemma shows the theoretical guarantees of Algorithm 3:

LEMMA 8. *Given that $\mathbb{E}[\frac{1}{KPT^*}] < \frac{12}{EPT}$, Algorithm 3 runs in $O(\ell(m+n)\log n/(\varepsilon')^2)$ expected time. In addition, it returns*

$KPT^+ \in [KPT^*, OPT]$ *with at least* $1 - n^{-\ell}$ *probability, if* $KPT^* \in [KPT/4, OPT]$.

Note that the time complexity of Algorithm 3 is smaller than that of Algorithm 1 by a factor of k, since the former only needs to accurately estimate the expected spread of one node set (i.e., S'_k), whereas the latter needs to ensure accurate estimations for $\binom{n}{k}$ node sets simultaneously.

We integrate Algorithm 3 into *TIM* and obtain an improved solution (referred to as *TIM*$^+$) as follows. Given G, k, ε, and ℓ, we first invoke Algorithm 2 to derive KPT^*. After that, we feed G, k, KPT^*, and a parameter ε' to Algorithm 3, and obtain KPT^+ in return. Then, we compute $\theta = \lambda / KPT^+$. Finally, we run Algorithm 1 with G, k, and θ as the input, and get the final result of influence maximization. It can be verified that when $\varepsilon' \geq \varepsilon/\sqrt{k}$, *TIM*$^+$ has the same time complexity with *TIM*, and it returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1-3n^{-\ell}$ probability. The success probability can be raised to $1 - n^{-\ell}$ by increasing ℓ by a factor of $1 + \log 3 / \log n$.

Finally, we discuss the choice of ε'. Ideally, we should set ε' to a value that minimizes the total number β of RR sets generated in Algorithms 1 and 3. However, β is difficult to estimate as it depends on unknown variables such as KPT^* and KPT^+. In our implementation of *TIM*$^+$, we set

$$\varepsilon' = 5 \cdot \sqrt[3]{\ell \cdot \varepsilon^2 / (k + \ell)}$$

for any $\varepsilon \leq 1$. This is obtained by using a function of ε' to roughly approximate β, and then taking the minimizer of the function.

4.2 Generalization to the Triggering Model

The *triggering model* [17] is a influence propagation model that generalizes the IC model. It assumes that each node v is associated with a *triggering distribution* $\mathcal{T}(v)$ over the power set of v's incoming neighbors, i.e., each sample from $\mathcal{T}(v)$ is a subset of the nodes that has an outgoing edge to v.

Given a seed set S, an influence propagation process under the triggering model works as follows. First, for each node v, we take a sample from $\mathcal{T}(v)$, and define the sample as the *triggering set* of v. After that, at timestamp 1, we activate the nodes in S. Then, at subsequent timestamp i, if an activated node appears in the triggering set of an inactive node v, then v becomes activated at timestamp $i+1$. The propagation process terminates when no more nodes can be activated.

The influence maximization problem under the triggering model asks for a size-k seed set S that can activate the largest number of nodes in expectation. To understand why the triggering model captures the IC model as a special case, consider that we assign a triggering distribution to each node v, such that each of v's incoming neighbors independently appears in v's trigger set with $p(e)$ probability, where e is the edge that goes from the neighbor to v. It can be verified that influence maximization under this distribution is equivalent to that under the IC model.

Interestingly, our solutions can be easily extended to support the triggering model. To explain, observe that Algorithms 1, 2, and 3 do not rely on anything specific to the IC model, except that they require a subroutine to generate random RR sets, whereas RR sets are defined under the IC model only. To address this issue, we revise the definition of RR sets to accommodate the triggering model, as explained in the following.

Suppose that we generate random graphs g from G, by first sampling a node set T for each node v from its triggering distribution $\mathcal{T}(v)$, and then removing any outgoing edge of v that does not point to a node in T. Let $\mathcal{G}$ be the distribution of g induced by the random choices of triggering sets. We refer to $\mathcal{G}$ as the *triggering graph distribution* for G. For any given node u and a graph g sampled from G, we define the reverse reachable (RR) set for u in g as the set of nodes that can reach u in g. In addition, we define a *random RR set* as one that is generated on an instance of g randomly sampled from $\mathcal{G}$, for a node selected from g uniformly at random.

To construct random RR sets defined above, we employ a randomized BFS algorithm as follows. Let v be a randomly selected node. Given v, we first take a sample T from v's triggering distribution $\mathcal{T}(v)$, and then put all nodes in T into a queue. After that, we iteratively extract the node at the top of the queue; for each node u extracted, we sample a set T' from u's triggering distribution, and we insert any unvisited node in T' into the queue. When the queue becomes empty, we terminate the process, and form a random RR set with the nodes visited during the process. The expected cost of the whole process is $O(EPT)$, where EPT denotes the expected number of edges in G that point to the nodes in a random RR set. This expected time complexity is the same as that of the algorithm for generating random RR sets under the IC model.

By incorporating the above BFS approach into Algorithms 1, 2, and 3, our solutions can readily support the triggering model. Our next step is to show that the revised solution retains the performance guarantees of *TIM* and *TIM*$^+$. For this purpose, we first present an extended version of Lemma 2 for the triggering model. (The proof of the lemma is almost identical to that of Lemma 2.)

LEMMA 9. *Let S be a fixed set of nodes, v be a fixed node, and $\mathcal{G}$ be the triggering graph distribution for G. Suppose that we generate an RR set R for v on a graph g sampled from $\mathcal{G}$. Let ρ_1 be the probability that S overlaps with R, and ρ_2 be the probability that S (as a seed set) can activate v in an influence propagation process on G under the triggering model. Then, $\rho_1 = \rho_2$.*

Next, we note that all of our theoretical analysis of *TIM* and *TIM*$^+$ is based on the Chernoff bounds and Lemma 2, without relying on any other results specific to the IC model. Therefore, once we establish Lemma 9, it is straightforward to combine it with the Chernoff bounds to show that, under the triggering model, both *TIM* and *TIM*$^+$ provide the same performance guarantees as in the case of the IC model. Thus, we have the following theorem:

THEOREM 3. *Under the triggering model, TIM (resp. TIM*$^+$*) runs in $O((k+\ell)(m+n)\log n/\varepsilon^2)$ expected time, and returns a $(1-1/e-\varepsilon)$-approximate solution with at least $1 - 2 \cdot n^{-\ell}$ probability (resp. $1 - 3 \cdot n^{-\ell}$ probability).*

5. THEORETICAL COMPARISONS

Comparison with *RIS*. Borgs et al. [3] show that, under the IC model, *RIS* can derive a $(1-1/e-\varepsilon)$-approximate solution for the influence maximization problem, with $O(k\ell^2(m+n)\log^2 n/\varepsilon^3)$ running time and at least $1 - n^{-\ell}$ success probability. The time complexity of *RIS* is larger than the expected time complexity of *TIM* and *TIM*$^+$ by a factor of $\ell \log n/\varepsilon$. Therefore, both *TIM* and *TIM*$^+$ are superior to *RIS* in terms of asymptotic performance.

Comparison with *Greedy*. As mentioned in Section 2.2, *Greedy* runs in $O(kmnr)$ time, where r is the number of Monte Carlo samples used to estimate the expected spread of each node set. Kempe et al. do not provide a formal result on how r should be set to achieve a $(1 - 1/e - \varepsilon)$-approximation ratio; instead, they only point out that when each estimation of expected spread has ε related error, *Greedy* returns a $(1 - 1/e - \varepsilon')$-approximate solution for a certain ε' [18].

We present a more detailed characterization on the relationship between r and *Greedy*'s approximation ratio:

Name	n	m	Type	Average degree
NetHEPT	15K	31K	undirected	4.1
Epinions	76K	509K	directed	13.4
DBLP	655K	2M	undirected	6.1
LiveJournal	4.8M	69M	directed	28.5
Twitter	41.6M	1.5G	directed	70.5

Table 2: Dataset characteristics.

LEMMA 10. Greedy *returns a* $(1 - 1/e - \varepsilon)$*-approximate solution with at least* $1 - n^{-\ell}$ *probability, if*

$$r \geq (8k^2 + 2k\varepsilon) \cdot n \cdot \frac{(\ell + 1)\log n + \log k}{\varepsilon^2 \cdot OPT}. \quad (10)$$

Assume that we know OPT in advance and set r to the smallest value satisfying the above inequality, in *Greedy*'s favor. In that case, the time complexity of *Greedy* is $O(k^3 \ell m n^2 \varepsilon^{-2} \log n / OPT)$. Given that $OPT \leq n$, this complexity is much worse than the expected time complexity of *TIM* and *TIM*$^+$.

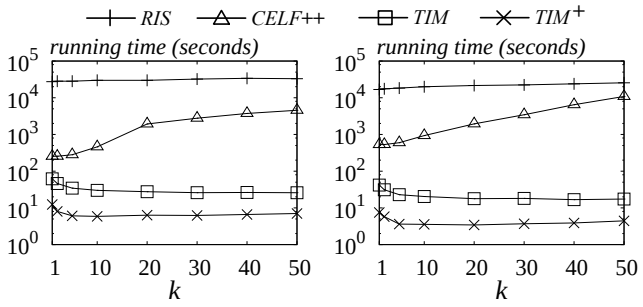
6. ADDITIONAL RELATED WORK

There has been a large body of literature on influence maximization over the past decade (see [2–7, 10, 16–19, 21, 28, 30, 31] and the references therein). Besides *Greedy* [17] and *RIS* [3], the work most related to ours is by Leskovec et al. [21], Chen et al. [6], and Goyal et al. [11]. In particular, Leskovec et al. [21] propose an algorithmic optimization of *Greedy* that avoids evaluating the expected spreads of a large number of node sets. This optimization reduces the computation cost of *Greedy* by up to 700-fold, without affecting its approximation guarantees. Subsequently, Chen et al. [6] and Goyal et al. [11] further enhance Leskovec et al.'s approach, and achieve up to 50% additional improvements in terms of efficiency.

Meanwhile, there also exist a plethora of algorithms [5–7, 10, 16, 19, 30, 31] that rely on heuristics to efficiently derive solutions for influence maximization. For example, Chen et al. [5] propose to reduce computation costs by omitting the social network paths with low propagation probabilities; Wang et al. [31] propose to divide the social network into smaller communities, and then identify influential nodes from each community individually; Goyal et al. [12] propose to estimate the expected spread of each node set S only based on the nodes that are close to S. In general, existing heuristic solutions are shown to be much more efficient than *Greedy* (and its aforementioned variants [6, 11, 21]), but they fail to retain the $(1 - 1/e - \varepsilon)$-approximation ratio. As a consequence, they tend to produce less accurate results, as shown in the experiments in [5–7, 10, 16, 19, 30, 31].

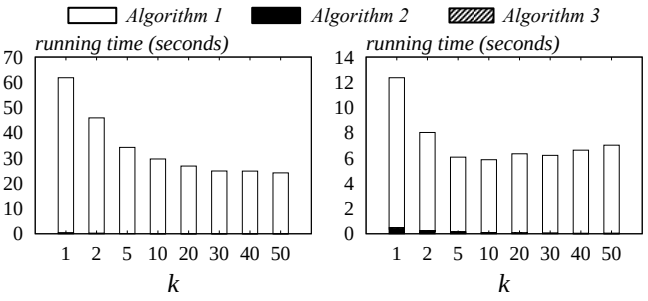
Considerable research has also been done to extend Kempe et al.'s formulation of influence maximization [17] to various new settings, e.g., when the influence propagation process follows a different model [10, 22], when there are multiple parties that compete with each other for social influence [2, 23], or when the influence propagation process terminates at a predefined timestamp [4]. The solutions derived for those scenarios are inapplicable under our setting, due to the differences in problem formulations. Finally, there is recent research on learning the parameters of influence propagation model (e.g., the propagation probability on each edge) from observed data [9, 20, 26]. This line of research complements (and is orthogonal to) the existing studies on influence maximization.

7. EXPERIMENTS



(a) The IC model. (b) The LT model.

Figure 3: Computation time vs. k on *NetHEPT*.



(a) *TIM* (the IC model). (b) *TIM*$^+$ (the IC model).

Figure 4: Breakdown of computation time on *NetHEPT*.

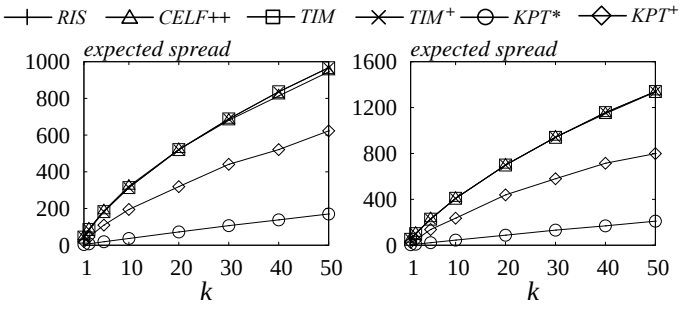
This section experimentally evaluates *TIM* and *TIM*$^+$. Our experiments are conducted on a machine with an Intel Xeon 2.4GHz CPU and 48GB memory, running 64bit Ubuntu 13.10. All algorithms tested are implemented in C++ and compiled with g++ 4.8.1.

7.1 Experimental Settings

Datasets. Table 2 shows the datasets used in our experiments. Among them, *NetHEPT*, *Epinions*, *DBLP*, and *LiveJournal* are benchmarks in the literature of influence maximization [19]. Meanwhile, *Twitter* contains a social network crawled from Twitter.com in July 2009, and it is publicly available from [1]. Note that *Twitter* is significantly larger than the other four datasets.

Propagation Models. We consider two influence propagation models, namely, the IC model (see Section 2.1) and the *linear threshold (LT)* model [17]. Specifically, the LT model is a special case of the triggering model, such that for each node v, any sample from v's triggering distribution $\mathcal{T}(v)$ is either $\emptyset$ or a singleton containing an incoming neighbor of v. Following previous work [7], we construct $\mathcal{T}(v)$ for each node v, by first assigning a random probability in $[0, 1]$ to each of v's incoming neighbors, and then normalizing the probabilities so that they sum up to 1. As for the IC model, we set the propagation probability of each edge e as follows: we first identify the node v that e points to, and then set $p(e) = 1/i$, where i denotes the in-degree of v. This setting of $p(e)$ is widely adopted in prior work [5, 10, 16, 30].

Algorithms. We compare our solutions with four methods, namely, *RIS* [3], *CELF++* [11], *IRIE* [16], and *SIMPATH* [12]. In particular, *CELF++* is a state-of-the-art variant of *Greedy* that considerably improves the efficiency of *Greedy* without affecting its theoretical guarantees, while *IRIE* and *SIMPATH* are the most advanced heuristic methods under the IC and LT models, respectively. We adopt the C++ implementations of *CELF++*, *IRIE*, and *SIMPATH* made available by their inventors, and we implement *RIS* and our



(a) The IC model. (b) The LT model.

Figure 5: Expected spreads, KPT^*, and KPT^+ on *NetHEPT*.

solutions in C++. Note that *RIS* is designed under the IC model only, but we incorporate the techniques in Section 4.2 into *RIS* and extend it to the LT model.

Parameters. Unless otherwise specified, we set $\varepsilon = 0.1$ and $k = 50$ in our experiments. For *RIS* and our solutions, we set ℓ in a way that ensures a success probability of $1 - 1/n$. For *CELF++*, we set the number of Monte Carlo steps to $r = 10000$, following the standard practice in the literature. Note that this choice of r is to the advantage of *CELF++* because, by Lemma 10, the value of r required in our experiments is always larger than 10000. In each of our experiments, we repeat each method three times and report the average result.

7.2 Comparison with *CELF++* and *RIS*

Our first set of experiments compares our solutions with *CELF++* and *RIS*, i.e., the state of the arts among the solutions that provide non-trivial approximation guarantees.

Results on *NetHEPT*. Figure 3 shows the computation cost of each method on the *NetHEPT* dataset, varying k from 1 to 50. Observe that *TIM*$^+$ consistently outperforms *TIM*, while *TIM* is up to two orders of magnitude faster than *CELF++* and *RIS*. In particular, when $k = 50$, *CELF++* requires more than an hour to return a solution, whereas *TIM*$^+$ terminates within ten seconds. These results are consistent with our theoretical analysis (in Section 5) that *Greedy*'s time complexity is much higher than those of *TIM* and *TIM*$^+$. On the other hand, *RIS* is the slowest method in all cases despite of its near-linear time complexity, because of the ε^{-3} term and the large hidden constant factor in its performance bound. One may improve the empirical efficiency of *RIS* by reducing the threshold τ on its running time (see Section 2.3), but in that case, the worst-case quality guarantee of *RIS* is not necessarily retained.

The computation overheads of *RIS* and *CELF++* increase with k, because (i) *RIS*'s threshold τ on running time is linear to k, while (ii) a larger k requires *CELF++* to evaluate the expected spread of an increased number of node sets. Surprisingly, when k increases, the running time of *TIM* and *TIM*$^+$ tends to decrease. To understand this, we show, in Figure 4, a breakdown of *TIM* and *TIM*$^+$'s computation overheads under the IC model. Evidently, both algorithms' overheads are mainly incurred by Algorithm 1, i.e., the node selection phase. Meanwhile, the computation cost of Algorithm 1 is mostly decided by the number θ of RR sets that it needs to generate. For *TIM*, we have $\theta = \lambda / KPT^*$, where λ is as defined in Equation 4, and KPT^* is a lower-bound of OPT produced by Algorithm 2. Both λ and KPT^* increase with k, and it happens that, on *NetHEPT*, the increase of KPT^* is more pronounced than that of λ, which leads to the decrease in *TIM*'s running time. Similar observations can be made on *TIM*$^+$ and on the case of the LT model.

From Figure 4, we can also observe that the computation cost of Algorithm 3 (i.e, the intermediate step) is negligible compared with the total cost of *TIM*$^+$. Yet, Algorithm 3 is so effective that it reduces *TIM*$^+$'s running time to at most $1/3$ of *TIM*'s. This indicates that Algorithm 3 returns a much tighter lower-bound of OPT than Algorithm 2 (i.e., the parameter estimation phase) does. To support this argument, Figure 5 illustrates the lower-bounds KPT^* and KPT^+ produced by Algorithms 2 and 3, respectively. Observe that KPT^+ is at least three times KPT^* in all cases, which is consistent with *TIM*$^+$'s 3-fold efficiency improvement over *TIM*.

In addition, Figure 5 also shows the expected spreads of the node sets selected by each method on *NetHEPT*. (We estimate the expected spread of a node set by taking the average of 10^5 Monte Carlo measurements.) There is no significant difference among the expected spreads pertinent to different methods.

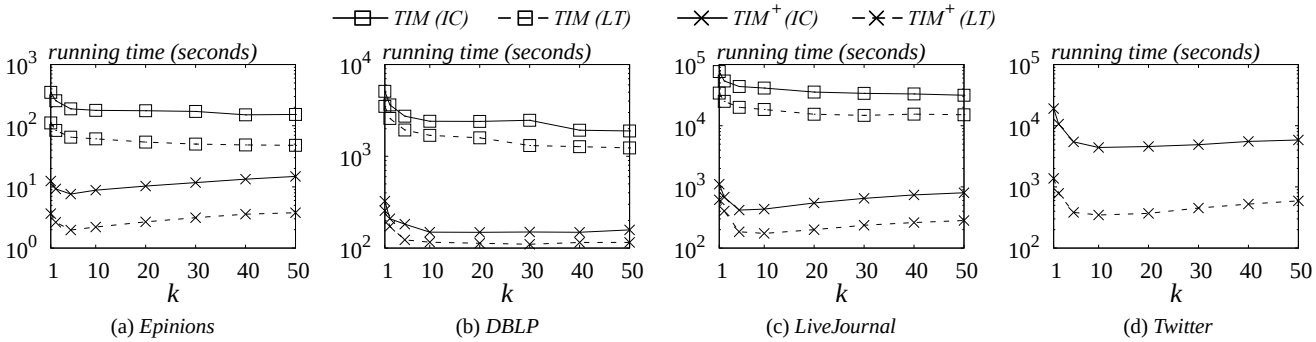
Results on Large Datasets. Next, we experiment with the four larger datasets, i.e., *Epinion*, *DBLP*, *LiveJournal*, and *Twitter*. As *RIS* and *CELF++* incur prohibitive overheads on those four datasets, we omit them from the experiments. Figure 6 shows the running time of *TIM* and *TIM*$^+$ on each dataset. Observe that *TIM*$^+$ outperforms *TIM* in all cases, by up to two orders of magnitude in terms of running time. Furthermore, even in the most adversarial case when $k = 1$, *TIM*$^+$ terminates within four hours under both the IC and LT models. (*TIM* is omitted from Figure 6d due to its excessive computation cost on *Twitter*.)

Interestingly, both *TIM* and *TIM*$^+$ are more efficient under the LT model than the IC model. This is caused by the fact that we use different methods to generate RR sets under the two models. Specifically, under the IC model, we construct each RR set with a randomized BFS on G; for each incoming edge that we encounter during the BFS, we need to generate a random number to decide whether the edge should be ignored. In contrast, when we perform a randomized BFS on G to create an RR set under the LT model, we generate a random number x for each node v that we visit, and we use x to pick an incoming edge of v to traverse. In other words, the number of random numbers required under the IC (resp. LT) model is proportional to the number of edges (resp. nodes) examined. Given that each of our datasets contains much more edges than nodes, it is not surprising that our solutions perform better under the LT model.

Finally, Figure 7 shows the running time of *TIM* and *TIM*$^+$ as a function of ε. The performance of both algorithms significantly improves with the increase of ε, since a larger ε leads to a less stringent requirement on the number of RR sets. In particular, when $\varepsilon \geq 0.2$, *TIM*$^+$ requires less than 1 hour to process *Twitter* under both the IC and LT models.

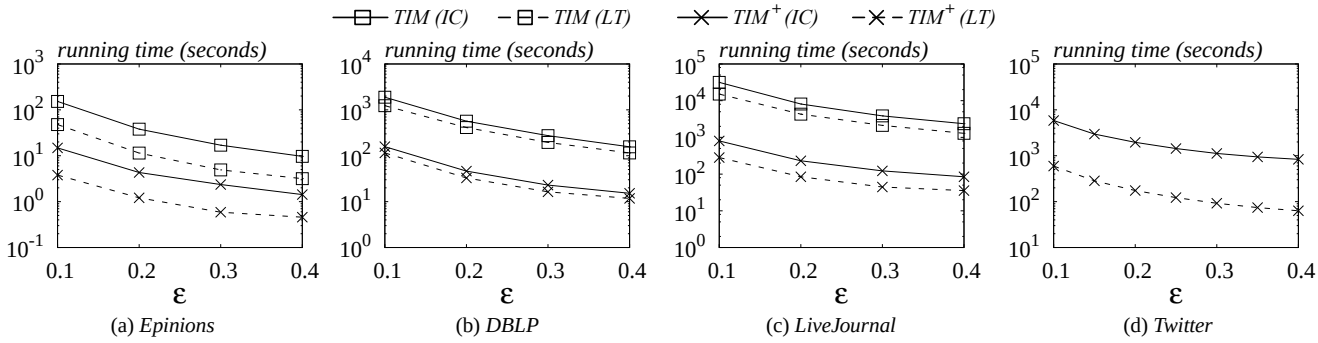
7.3 Comparison with *IRIE* and *SIMPATH*

Our second set of experiments compares *TIM*$^+$ with *IRIE* [16] and *SIMPATH* [12], namely, the state-of-the-art heuristic methods under the IC and LT models, respectively. (We omit *TIM* as it performs consistently worse than *TIM*$^+$.) Both *IRIE* and *SIMPATH* have two internal parameters that control the trade-off between computation cost and result accuracy. In our experiments, we set those parameters according to the recommendations in [12, 16]. Specifically, we set *IRIE* parameters α and θ to 0.7 and $1/320$, respectively, and *SIMPATH*'s parameters η and ℓ to 10^{-3} and 4, respectively. For *TIM*$^+$, we set $\varepsilon = \ell = 1$, in which case *TIM*$^+$ provides weak theoretical guarantees but high empirical efficiency. We evaluate the algorithms on all datasets except *Twitter*, as the



(a) *Epinions* (b) *DBLP* (c) *LiveJournal* (d) *Twitter*

Figure 6: Running time vs. k on large datasets.



(a) *Epinions* (b) *DBLP* (c) *LiveJournal* (d) *Twitter*

Figure 7: Running time vs. ε on large datasets.

memory consumptions of *IRIE* and *SIMPATH* exceed the size of the memory on our testing machine (i.e., 48GB).

Figure 8 shows the running time of *TIM*$^+$ and *IRIE* under the IC model, varying k from 1 to 50. The computation cost of *TIM*$^+$ tends to decrease with the increase of k, as a result of the subtle interplay among several variables (e.g., λ, KPT^*, and KPT^+) that decide the number of random RR sets required in *TIM*$^+$. Meanwhile, *IRIE*'s computation time increases with k, since (i) it adopts a greedy approach to iteratively select k nodes from the input graph G, and (ii) a larger k results in more iterations in *IRIE*, which leads to a higher processing cost. Overall, *TIM*$^+$ is not as efficient as *IRIE* when k is small, but it clearly outperforms *IRIE* on all datasets when $k > 20$. In particular, when $k = 50$, *TIM*$^+$'s computation time on *LiveJournal* is less than 5% of *IRIE*'s.

Figure 9 illustrates the expected spreads of the node sets returned by *TIM*$^+$ and *IRIE*. Compared with *IRIE*, *TIM*$^+$ have (i) noticeably higher expected spreads on *DBLP* and *LiveJournal*, and (ii) similar expected spreads on *NetHEPT* and *Epinion*. This indicates that *TIM*$^+$ generally provides more accurate results than *IRIE* does, even when we set $\varepsilon = \ell = 1$ for *TIM*$^+$.

Figure 10 compares the computation efficiency of *TIM*$^+$ and *SIMPATH* under the LT model, when k varies. Observe that *TIM*$^+$ consistently outperforms *SIMPATH* by large margins. In particular, when $k = 50$, the former's running time on *LiveJournal* is lower than the latter's by three orders of magnitude. Furthermore, as shown in Figure 11, *TIM*$^+$'s expected spreads are significantly higher than *SIMPATH*'s on *LiveJournal*, and are no worse on the other three datasets. Therefore, *TIM*$^+$ is clearly more preferable than *SIMPATH* for influence maximization under the LT model.

7.4 Memory Consumptions

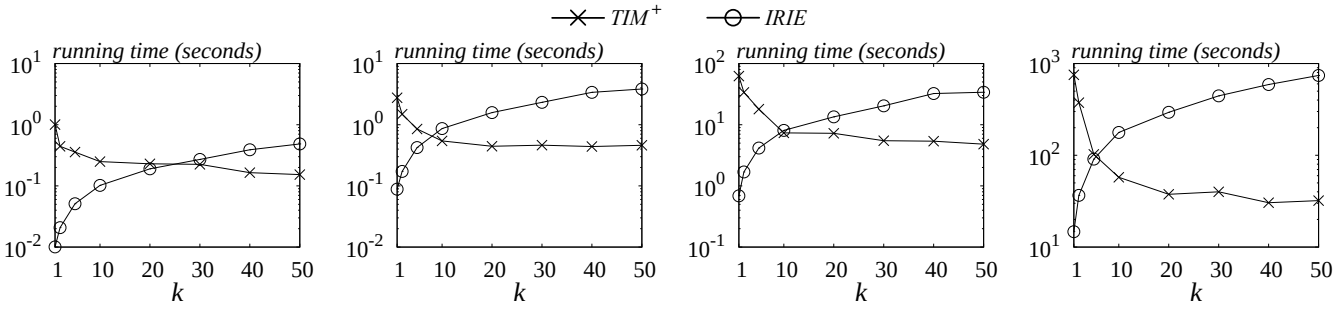
Our last set of experiments evaluates *TIM*$^+$'s memory consumptions, setting $\varepsilon = 0.1$ and $\ell = 1 + \log 3/\log n$ (i.e., we ensure a success probability of at least $1 - 1/n$). Note that $\varepsilon = 0.1$ is adversarial to *TIM*$^+$, due to the following reasons:

1. The memory costs of *TIM*$^+$ is mainly incurred by the set $\mathcal{R}$ of random RR sets generated in Algorithm 1;
2. *TIM*$^+$ sets the size of $\mathcal{R}$ to λ/KPT^+, where λ is as defined in Equation 4 and KPT^+ is a lowerbound of OPT generated by Algorithm 3;
3. λ is inverse proportional to ε^2, i.e., a smaller ε leads to a larger $\mathcal{R}$, which results in a higher space overhead.

Figure 12 shows the memory costs of *TIM*$^+$ on each dataset under the IC and LT models. In all cases, *TIM*$^+$ requires more memory under the IC model than under the LT model. The reason is that $\mathcal{R}$'s size is inverse proportional to KPT^+, while KPT^+ tends to be larger under the LT model (see Figure 5 for example). The memory consumption of *TIM*$^+$ tends to be larger when the dataset size increases, since $\mathcal{R} = \lambda/KPT^+$, while λ increases with n, i.e., the number of nodes in the dataset. But interestingly, *TIM*$^+$ incurs a higher space overhead on *NetHEPT* than on *Epinion*, even though the latter has a larger number of nodes. To explain, observe from Figures 9 and 11 that nodes in *Epinion* tend to have much higher expected spreads than those in *NetHEPT*. As a consequence, *TIM*$^+$ obtains a considerably larger KPT^+ from *Epinion* than from *NetHEPT*. This pronounced increase in KPT^+ renders $\mathcal{R} = \lambda/KPT^+$ smaller on *Epinion* than on *NetHEPT*, despite of the fact that λ is smaller on the latter.

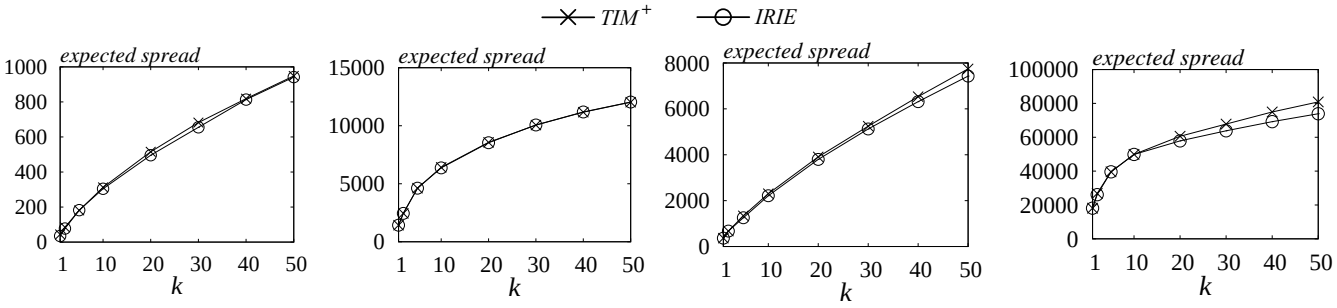
8. CONCLUSION

This paper presents *TIM*, an influence maximization algorithm that supports the triggering model by Kempe et al. [17]. The algorithm runs in $O((k + \ell)(n + m) \log n/\varepsilon^2)$ expected time, and returns $(1 - 1/e - \varepsilon)$-approximate solutions with at least $1 - n^{-\ell}$



(a) *NetHEPT* (b) *Epinions* (c) *DBLP* (d) *LiveJournal*

Figure 8: Running time vs. k under the IC model.



(a) *NetHEPT* (b) *Epinions* (c) *DBLP* (d) *LiveJournal*

Figure 9: Expected spreads vs. k under the IC model.

probability. In addition, it incorporates heuristic optimizations that lead to up to 100-fold improvements in empirical efficiency. Our experiments show that, when $k = 50$, $\varepsilon = 0.2$, and $\ell = 1$, the algorithm can process a billion-edge graph on a commodity machine within an hour. Such practical efficiency is unmatched by any existing solutions that provide non-trivial approximation guarantees for the influence maximization problem. For future work, we plan to investigate how we can turn *TIM* into a distributed algorithm, so as to handle massive graphs that do not fit in the main memory of a single machine. In addition, we plan to extend *TIM* to other formulations of the influence maximization problem, e.g., *competitive influence maximization* [2, 23].

9. REFERENCES

[1] `http://an.kaist.ac.kr/traces/WWW2010.html`.

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

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

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

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

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

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

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

[9] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, pages 241–250, 2010.

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

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

[12] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *ICDM*, pages 211–220, 2011.

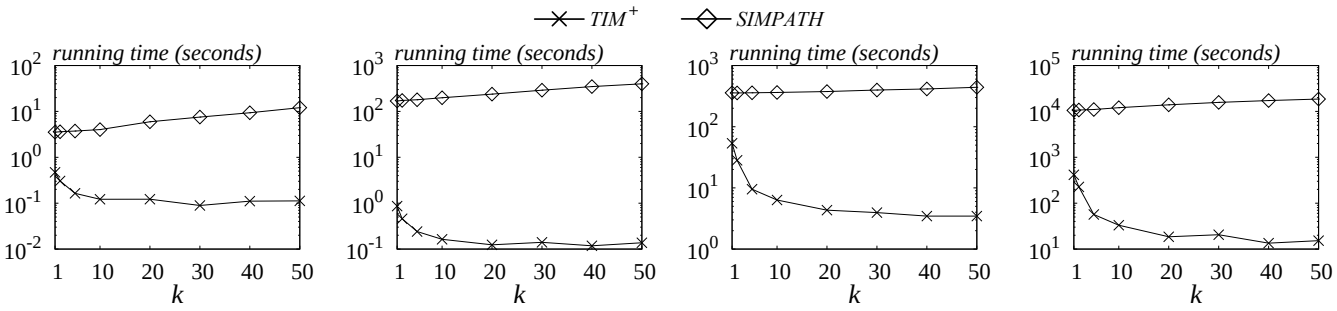
[13] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

[14] E. M. J. Goldenberg, B. Libai. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

[15] E. M. J. Goldenberg, B. Libai. Using complex systems analysis to advance marketing theory development. *American Journal of Sociology*, 9:1, 2001.

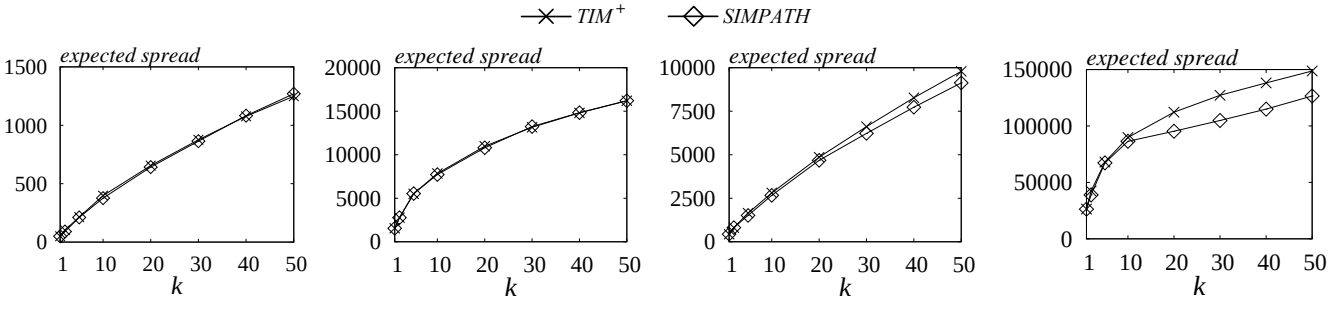
[16] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *ICDM*, pages 918–923, 2012.

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



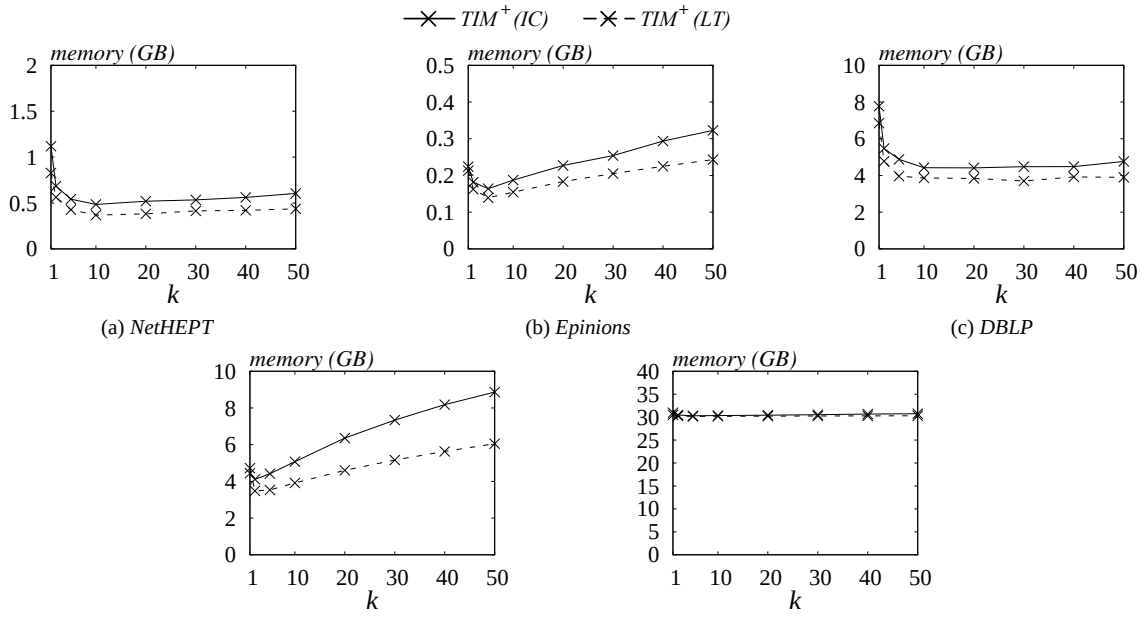
(a) *NetHEPT* (b) *Epinions* (c) *DBLP* (d) *LiveJournal*

Figure 10: Running time vs. k under the LT model.



(a) *NetHEPT* (b) *Epinions* (c) *DBLP* (d) *LiveJournal*

Figure 11: Expected spreads vs. k under the LT model.



(a) *NetHEPT* (b) *Epinions* (c) *DBLP*

(d) *LiveJournal* (e) *Twitter*

Figure 12: Memory consumptions of *TIM*$^+$ vs. k.

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

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

[20] K. Kutzkov, A. Bifet, F. Bonchi, and A. Gionis. Strip: stream learning of influence probabilities. In *KDD*, pages 275–283, 2013.

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

[22] Y. Li, W. Chen, Y. Wang, and Z.-L. Zhang. Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships. In *WSDM*, pages 657–666, 2013.

[23] W. Lu, F. Bonchi, A. Goyal, and L. V. S. Lakshmanan. The bang for the buck: fair competitive viral marketing from the host perspective. In *KDD*, pages 928–936, 2013.

[24] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

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

[26] K. Saito, N. Mutoh, T. Ikeda, T. Goda, and K. Mochizuki. Improving search efficiency of incremental variable selection by using second-order optimal criterion. In *KES (3)*, pages 41–49, 2008.

[27] T. Schelling. *Micromotives and Macrobehavior*. W. W. Norton & Company, 2006.

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

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

[30] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Min. Knowl. Discov.*, 25(3):545–576, 2012.

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

APPENDIX

Proof of Lemma 2. Let g be a graph constructed from G by removing each edge e with $1 - p(e)$ probability. Then, ρ_2 equals the probability that v is reachable from S in g. Meanwhile, by Definition 1, ρ_1 equals the probability that g contains a directed path that ends at v and starts at a node in S. It follows that $\rho_1 = \rho_2$. □

Proof of Corollary 1. Observe that $\mathbb{E}[F_{\mathcal{R}}(S)]$ equals the probability that S intersects a random RR set, while $\mathbb{E}[I(S)]/n$ equals the probability that a randomly selected node can be activated by S in an influence propagation process on G. By Lemma 2, the two probabilities are equal, leading to $\mathbb{E}[n \cdot F_{\mathcal{R}}(S)] = \mathbb{E}[I(S)]$. □

Proof of Lemma 3. Let ρ be the probability that S overlaps with a random RR set. Then, $\theta \cdot F_{\mathcal{R}}(S)$ can be regarded as the sum of θ i.i.d. Bernoulli variables with a mean ρ. By Corollary 1,

$$\rho = \mathbb{E}[F_{\mathcal{R}}(S)] = \mathbb{E}[I(S)]/n.$$

Then, we have

$$\begin{aligned}
&\Pr\left[\left|n \cdot F_{\mathcal{R}}(S) - \mathbb{E}[I(S)]\right| \geq \frac{\varepsilon}{2} \cdot OPT\right] \\
&= \Pr\left[\left|\theta \cdot F_{\mathcal{R}}(S) - \rho\theta\right| \geq \frac{\varepsilon\theta}{2n} \cdot OPT\right] \\
&= \Pr\left[\left|\theta \cdot F_{\mathcal{R}}(S) - \rho\theta\right| \geq \frac{\varepsilon \cdot OPT}{2n\rho} \cdot \rho\theta\right]. \qquad (11)
\end{aligned}$$

Let $\delta = \varepsilon \cdot OPT/(2n\rho)$. By the Chernoff bounds, Equation 2, and the fact that $\rho = \mathbb{E}[I(S)]/n \leq OPT/n$, we have

$$\begin{aligned}
\text{r.h.s. of Eqn. 11} &< 2\exp\left(-\frac{\delta^2}{2+\delta} \cdot \rho\theta\right) \\
&= 2\exp\left(-\frac{\varepsilon^2 \cdot OPT^2}{8n^2\rho + 2\varepsilon n \cdot OPT} \cdot \theta\right) \\
&\leq 2\exp\left(-\frac{\varepsilon^2 \cdot OPT^2}{8n \cdot OPT + 2\varepsilon n \cdot OPT} \cdot \theta\right) \\
&= 2\exp\left(-\frac{\varepsilon^2 \cdot OPT}{(8+2\varepsilon) \cdot n} \cdot \theta\right) \;\leq \frac{1}{\binom{n}{k} \cdot n^l}.
\end{aligned}$$

Therefore, the lemma is proved. □

Proof of Theorem 1. Let S_k be the node set returned by Algorithm 1, and S_k^+ be the size-k node set that maximizes $F_{\mathcal{R}}(S_k^+)$ (i.e., S_k^+ covers the largest number of RR sets in $\mathcal{R}$). As S_k is derived from $\mathcal{R}$ using a $(1-1/e)$-approximate algorithm for the maximum coverage problem, we have $F_{\mathcal{R}}(S_k) \geq (1-1/e) \cdot F_{\mathcal{R}}(S_k^+)$. Let S_k° be the optimal solution for the influence maximization problem on G, i.e., $\mathbb{E}[I(S_k^\circ)] = OPT$. We have $F_{\mathcal{R}}(S_k^+) \geq F_{\mathcal{R}}(S_k^\circ)$, which leads to $F_{\mathcal{R}}(S_k) \geq (1-1/e) \cdot F_{\mathcal{R}}(S_k^\circ)$.

Assume that θ satisfies Equation 2. By Lemma 3, Equation 3 holds with at least $1 - n^{-\ell}/\binom{n}{k}$ probability for any given size-k node set S. The, by the union bound, Equation 3 should hold simultaneously for all size-k node sets with at least $1 - n^{-\ell}$ probability. In that case, we have

$$\begin{aligned}
\mathbb{E}[I(S_k)] &> n \cdot F_{\mathcal{R}}(S_k) - \varepsilon/2 \cdot OPT \\
&\geq (1-1/e) \cdot n \cdot F_{\mathcal{R}}(S_k^+) - \varepsilon/2 \cdot OPT \\
&\geq (1-1/e) \cdot n \cdot F_{\mathcal{R}}(S_k^\circ) - \varepsilon/2 \cdot OPT \\
&\geq (1-1/e) \cdot (1-\varepsilon/2) \cdot OPT - \varepsilon/2 \cdot OPT \\
&> (1-1/e-\varepsilon) \cdot OPT.
\end{aligned}$$

Thus, the theorem is proved. □

Proof of Lemma 4. Let R be a random RR set, p_R be the probability that a randomly selected edge from G points to a node in R. Then, $EPT = \mathbb{E}[p_R \cdot m]$, where the expectation is taken over the random choices of R.

Let v^* be a sample from $\mathcal{V}^*$, and $b(v^*, R)$ be a boolean function that returns 1 if $v^* \in R$, and 0 otherwise. Then, for any fixed R,

$$p_R = \sum_{v^*} \big(\Pr[v^*] \cdot b(v^*, R)\big).$$

Now consider that we fix v^* and vary R. Define

$$p_{v^*} = \sum_{R} \big(\Pr[R] \cdot b(v^*, R)\big).$$

By Lemma 2, p_{v^*} equals the probability that a randomly selected node can be activated in an influence propagation process when $\{v^*\}$ is used as the seed set. Therefore, $\mathbb{E}[p_{v^*}] = \mathbb{E}[I(\{v^*\})]/n$. This leads to

$$\begin{aligned}
EPT/m = \mathbb{E}[p_R] &= \sum_{R} \big(\Pr[R] \cdot p_R\big) \\
&= \sum_{R} \Big(\Pr[R] \cdot \sum_{v^*} \big(\Pr[v^*] \cdot b(v^*, R)\big)\Big) \\
&= \sum_{v^*} \Big(\Pr[v^*] \cdot \sum_{R} \big(\Pr[R] \cdot b(v^*, R)\big)\Big) \\
&= \sum_{v^*} \big(\Pr[v^*] \cdot p_{v^*}\big) = \mathbb{E}[p_{v^*}] = \mathbb{E}[I(\{v^*\})]/n.
\end{aligned}$$

Thus, the lemma is proved. □

Proof of Lemma 5. Let S^* be a node set formed by k samples from $\mathcal{V}^*$, with duplicates removed. Let R be a random RR set, and α_R be the probability that S^* overlaps with R. Then, by Corollary 1,

$$KPT = \mathbb{E}[I(S^*)] = \mathbb{E}[n \cdot \alpha_R].$$

Consider that we sample k times over a uniform distribution on the edges in G. Let E^* be the set of edges sampled, with duplicates removed. Let α'_R be the probability that one of the edges in E^* points to a node in R. It can be verified that $\alpha'_R = \alpha_R$. Furthermore, given that there are $w(R)$ edges in G that point to nodes in R, $\alpha'_R = 1 - (1 - w(R)/m)^k = \kappa(R)$. Therefore,

$$KPT = \mathbb{E}[n \cdot \alpha_R] = \mathbb{E}[n \cdot \alpha'_R] = \mathbb{E}\left[n \cdot \kappa(R)\right],$$

which proves the lemma. □

Proof of Lemma 6. Let $\delta = (2^{-i} - \mu)/\mu$. By the Chernoff bounds,

$$\begin{aligned}\Pr\left[\frac{s_i}{c_i} > 2^{-i}\right] &\leq \exp\left(-\frac{\delta^2}{2+\delta} \cdot c_i \cdot \mu\right) \\ &= \exp\left(-c_i \cdot (2^{-i} - \mu)^2/(2^{-i} + \mu)\right) \\ &\leq \exp\left(-c_i \cdot 2^{-i-1}/3\right) \quad = \frac{1}{n^\ell \cdot \log_2 n}.\end{aligned}$$

This completes the proof. □

Proof of Lemma 7. Let $\delta = (\mu - 2^{-i})/\mu$. By the Chernoff bounds,

$$\begin{aligned}\Pr\left[\frac{s_i}{c_i} \leq 2^{-i}\right] &\leq \exp\left(-\frac{\delta^2}{2} \cdot c_i \cdot \mu\right) \\ &= \exp\left(-c_i \cdot (\mu - 2^{-i})^2/(2 \cdot \mu)\right) \\ &\leq \exp\left(-c_i \cdot \mu/8\right) \quad < n^{-\ell \cdot 2^{i-j-1}}/\log_2 n.\end{aligned}$$

This completes the proof. □

Proof of Theorem 2. Assume that $KPT/n \in [2^{-j}, 2^{-j+1}]$. We first prove the accuracy of the KPT^* returned by Algorithm 2.

By Lemma 6 and the union bound, Algorithm 2 terminates in or before the $(j-2)$-th iteration with less than $n^{-\ell}(j-2)/\log_2 n$ probability. On the other hand, if Algorithm 2 reaches the $(j+1)$-th iteration, then by Lemma 7, it terminates in the $(j+1)$-th iteration with at least $1 - n^{-\ell}/\log_2 n$ probability. Given the union bound and the fact that Algorithm 2 has at most $\log_2 n - 1$ iterations, Algorithm 2 should terminate in the $(j-1)$-th, j-th, or $(j+1)$-th iteration with a probability at least $1 - n^{-\ell}(\log_2 n - 2)/\log_2 n$. In that case, KPT^* must be larger than $n/2 \cdot 2^{-j-1}$, which leads to $KPT^* > KPT/4$. Furthermore, KPT^* should be $n/2$ times the average of at least c_{j-1} i.i.d. samples from $\mathcal{K}$. By the Chernoff bounds, it can be verified that

$$\Pr\left[KPT^* \geq KPT\right] \leq n^{-\ell}/\log_2 n.$$

By the union bound, Algorithm 2 returns, with at least $1 - n^{-\ell}$ probability, $KPT^* \in [KPT/4, KPT] \subseteq [KPT/4, OPT]$.

Next, we analyze the expected running time of Algorithm 2. Recall that the i-th iteration of the algorithm generates c_i RR sets, and each RR sets takes $O(EPT)$ expected time. Given that $c_{i+1} = 2 \cdot c_i$ for any i, the first $j+1$ iterations generate less than $2 * c_{j+1}$ RR sets in total. Meanwhile, for any $i' \geq j+2$, Lemma 7 shows that Algorithm 2 has at most $n^{-\ell \cdot 2^{i'-j-1}}/\log_2 n$ probability to reach the i'-th iteration. Therefore, when $n \geq 2$ and $\ell \geq 1/2$, the expected number of RR sets generated after the first $j+1$ iterations is less than

$$\sum_{i'=j+2}^{\log_2 n - 1} \left(c_{i'} \cdot n^{-\ell \cdot 2^{i'-j-1}}/\log_2 n\right) \quad < \quad c_{j+2}.$$

Hence, the expected total number of RR sets generated by Algorithm 2 is less than $2c_{j+1} + c_{j+2} = 2c_{j+2}$. Therefore, the expected time complexity of the algorithm is

$$\begin{aligned}O(c_{j+2} \cdot EPT) &= O(2^j \ell \log n \cdot EPT) \\ &= O(2^j \ell \log n \cdot (1 + \frac{m}{n}) \cdot KPT) \\ &= O(2^j \ell \log n \cdot (m+n) \cdot 2^{-j}) = O(\ell(m+n)\log n).\end{aligned}$$

Finally, we show that $\mathbb{E}[1/KPT^*] < 12/KPT$. Observe that if Algorithm 2 terminates in the i-th iteration, it returns $KPT^* \geq n \cdot 2^{-i-1}$. Let ζ_i denote the event that Algorithm 2 stops in the i-th iteration. By Lemma 7, when $n \geq 2$ and $\ell \geq 1/2$, we have

$$\begin{aligned}\mathbb{E}[1/KPT^*] &= \textstyle\sum_{i=1}^{\log_2 n - 1} \left(2^{i+1}/n \cdot \Pr[\zeta_i]\right) \\ &< \textstyle\sum_{i=j+2}^{\log_2 n - 1} \left(2^{i+1}/n \cdot \left(n^{-\ell \cdot 2^{i-j-1}}/\log_2 n\right)\right) + 2^{j+2}/n \\ &< (2^{j+3} + 2^{j+2})/n \quad \leq 12/KPT.\end{aligned}$$

This completes the proof. □

Proof of Lemma 8. We first analyze the expected time complexity of Algorithm 3. Observe that Lines 1-6 in Algorithm 3 run in time linear the total size of the RR sets in $\mathcal{R}'$, i.e., the set of all RR sets generated in the last iteration of Algorithm 2. Given that Algorithm 2 has an $O(\ell(m+n)\log n)$ expected time complexity (see Theorem 2), the expected total size of the RR sets in $\mathcal{R}'$ should be no more than $O(\ell(m+n)\log n)$. Therefore, Lines 1-6 of Algorithm 3 have an expected time complexity $O(\ell(m+n)\log n)$.

On the other hand, the expected time complexity of Lines 7-12 of Algorithm 3 is $O\left(\mathbb{E}\left[\frac{\lambda'}{KPT^*}\right] \cdot EPT\right)$, since they generate $\frac{\lambda'}{KPT^*}$ random RR sets, each of which takes $O(EPT)$ expected time. By Theorem 2, $\mathbb{E}[\frac{1}{KPT^*}] < \frac{12}{KPT}$. In addition, by Equation 7, $EPT \leq \frac{m}{n} KPT$. Therefore,

$$\begin{aligned}O\left(\mathbb{E}\left[\tfrac{\lambda'}{KPT^*}\right] \cdot EPT\right) &= O\left(\tfrac{\lambda'}{KPT} \cdot EPT\right) \\ &= O\left(\tfrac{\lambda'}{KPT} \cdot (1 + \tfrac{m}{n}) \cdot KPT\right) \\ &= O\left(\ell(m+n)\log n/(\varepsilon')^2\right).\end{aligned}$$

Therefore, the expected time complexity of Algorithm 3 is $O\left(\ell(m+n)\log n/(\varepsilon')^2\right)$.

Next, we prove that Algorithm 3 returns $KPT^+ \in [KPT^*, OPT]$ with a high probability. First, observe that $KPT^+ \geq KPT^*$ trivially holds, as Algorithm 3 sets $KPT^+ = \max\{KPT', KPT^*\}$, where KPT' is derived in Line 11 of Algorithm 3. To show that $KPT^+ \in [KPT^*, OPT]$, it suffices to prove that $KPT' \leq OPT$.

By Line 11 of Algorithm 3, $KPT' = f \cdot n/(1+\varepsilon')$, where f is the fraction of RR sets in $\mathcal{R}''$ that is covered by S'_k, while $\mathcal{R}''$ is a set of θ' random RR sets, and S'_k is a size-k node set generated from Lines 1-6 in Algorithm 3. Therefore, $KPT' \leq OPT$ if and only if $f \cdot n \leq (1+\varepsilon') \cdot OPT$.

Let ρ' be the probability that a random RR set is covered by S'_k. By Corollary 1, $\rho' = \mathbb{E}[I(S'_k)]/n$. In addition, $f \cdot \theta'$ can be regarded as the sum of θ' i.i.d. Bernoulli variables with a mean ρ'. Therefore, we have

$$\begin{aligned}&\Pr\left[f \cdot n > (1+\varepsilon') \cdot OPT\right] \\ &\leq \Pr\left[n \cdot f - \mathbb{E}\left[I(S'_k)\right] > \varepsilon' \cdot OPT\right] \\ &= \Pr\left[\theta' \cdot f - \theta' \cdot \rho' > \frac{\theta'}{n} \cdot \varepsilon' \cdot OPT\right] \\ &= \Pr\left[\theta' \cdot f - \theta' \cdot \rho' > \frac{\varepsilon' \cdot OPT}{n \cdot \rho'} \cdot \theta' \cdot \rho'\right] \qquad (12)\end{aligned}$$

let $\delta = \varepsilon' \cdot OPT/(n\rho')$. By the Chernoff bounds, we have

$$\begin{aligned}
\text{r.h.s. of Eqn. 12} &\leq \exp\left(-\frac{\delta^2}{2+\delta}\cdot\rho'\theta'\right)\\
&= \exp\left(-\frac{\varepsilon'^2\cdot OPT^2}{2n^2\rho' + \varepsilon' n\cdot OPT}\cdot\theta'\right)\\
&\leq \exp\left(-\frac{\varepsilon'^2\cdot OPT^2}{2n\cdot OPT + \varepsilon' n\cdot OPT}\cdot\theta'\right)\\
&= \exp\left(-\frac{\varepsilon'^2\cdot OPT}{(2+\varepsilon')\cdot n}\cdot\frac{\lambda'}{KPT^*}\right)\\
&\leq \exp\left(-\frac{\varepsilon'^2\cdot\lambda'}{(2+\varepsilon')\cdot n}\right) \leq \frac{1}{n^l}.
\end{aligned}$$

Therefore, $KPT' = f\cdot n/(1+\varepsilon') \leq OPT$ holds with at least $1-n^{-l}$ probability. This completes the proof. □

Proof of Lemma 9. Let g be a graph constructed from G by first sampling a node set T for each node v from its triggering distribution $\mathcal{T}(v)$, and then removing any outgoing edge of v that does not point to a node in T. Then, ρ_2 equals the probability that v is reachable from S in g. Meanwhile, by the definition of RR sets under the triggering model, ρ_1 equals the probability that g contains a directed path that ends at v and starts at a node in S. It follows that $\rho_1 = \rho_2$. □

Proof of Lemma 10. Let S be any node set that contains no more than k nodes in G, and $\xi(S)$ be an estimation of $\mathbb{E}[I(S)]$ using r Monte Carlo steps. We first prove that, if r satisfies Equation 10, then $\xi(S)$ will be close to $\mathbb{E}[I(S)]$ with a high probability.

Let $\mu = \mathbb{E}[I(S)]/n$ and $\delta = \varepsilon OPT/(2kn\mu)$. By the Chernoff bounds, we have

$$\begin{aligned}
&\Pr\left[|\xi(S) - \mathbb{E}[I(S)]| > \frac{\varepsilon}{2k}OPT\right]\\
&= \Pr\left[\left|r\cdot\frac{\xi(S)}{n} - r\cdot\frac{\mathbb{E}[I(S)]}{n}\right| > \frac{\varepsilon}{2kn}\cdot r\cdot OPT\right]\\
&= \Pr\left[\left|r\cdot\frac{\xi(S)}{n} - r\cdot\frac{\mathbb{E}[I(S)]}{n}\right| > \delta\cdot r\cdot\mu\right]\\
&< 2\exp\left(-\frac{\delta^2}{2+\delta}\cdot r\cdot\mu\right)\\
&= 2\exp\left(-\frac{\varepsilon^2}{(8k^2+2k\varepsilon)\cdot n}\cdot r\cdot\mu\right)\\
&= 2\exp\left((\ell+1)\log n + \log k\right)\\
&= \frac{1}{k\cdot n^{\ell+1}} \qquad (13)
\end{aligned}$$

Observe that, given G and k, *Greedy* runs in k iterations, each of which estimates the expected spreads of at most n node sets with sizes no more than k. Therefore, the total number of node sets inspected by *Greedy* is at most kn. By Equation 13 and the union bound, with at least $1-n^{-\ell}$ probability, we have

$$\left|\xi(S') - \mathbb{E}[I(S')]\right| \leq \frac{\varepsilon}{2k}OPT, \qquad (14)$$

for all those kn node sets S' simultaneously. In what follows, we analyze the accuracy of *Greedy*'s output, under the assumption that for any node set S' considered by *Greedy*, it obtain a sample of $\xi(S')$ that satisfies Equation 14. For convenience, we abuse notation and use $\xi(S')$ to denote the aforementioned sample.

Let $S_0 = \emptyset$, and S_i ($i \in [1,k]$) be the node set selected by *Greedy* in the i-th iteration. We define $x_i = OPT - I(S_i)$, and $y_i(v) = I\left(S_{i-1}\cup\{v\}\right) - I(S_{i-1})$ for any node v. Let v_i be the node that maximizes $y_i(v_i)$. Then, $y_i(v_i) \geq x_{i-1}/k$ must hold; otherwise, for any size-k node S, we have

$$\begin{aligned}
I(S) &\leq I(S_{i-1}) + I(S\setminus S_{i-1})\\
&\leq I(S_{i-1}) + k\cdot y_i(v_i)\\
&< I(S_{i-1}) + x_{i-1} = OPT,
\end{aligned}$$

which contradicts the definition of OPT.

Recall that, in each iteration of *Greedy*, it adds into S_{i-1} the node v that leads to the largest $\xi(S_{i-1}\cup\{v\})$. Therefore,

$$\xi(S_i) - \xi(S_{i-1}) \geq \xi(S_{i-1}\cup\{v_i\}) - \xi(S_{i-1}). \qquad (15)$$

Combining Equations 14 and 15, we have

$$\begin{aligned}
&x_{i-1} - x_i\\
&= I(S_i) - I(S_{i-1})\\
&\geq \xi(S_i) - \frac{\varepsilon}{2k}OPT - \xi(S_{i-1}) + \Big(\xi(S_{i-1}) - I(S_{i-1})\Big)\\
&\geq \xi(S_{i-1}\cup\{v_i\}) - \xi(S_{i-1}) - \frac{\varepsilon}{2k}OPT\\
&\quad + \Big(\xi(S_{i-1}) - I(S_{i-1})\Big)\\
&\geq I\Big(S_{i-1}\cup\{v_i\}\Big) - I(S_{i-1}) - \frac{\varepsilon}{k}OPT\\
&\geq \frac{1}{k}x_{i-1} - \frac{\varepsilon}{k}OPT. \qquad (16)
\end{aligned}$$

Equation 16 leads to

$$\begin{aligned}
x_k &\leq \left(1-\frac{1}{k}\right)\cdot x_{k-1} + \frac{\varepsilon}{k}OPT\\
&\leq \left(1-\frac{1}{k}\right)^2\cdot x_{k-2} + \left(1+\left(1-\frac{1}{k}\right)\right)\cdot\frac{\varepsilon}{k}OPT\\
&\leq \left(1-\frac{1}{k}\right)^k\cdot x_0 + \sum_{i=0}^{k-1}\left(\left(1-\frac{1}{k}\right)^i\cdot\frac{\varepsilon}{k}OPT\right)\\
&= \left(1-\frac{1}{k}\right)^k\cdot OPT + \left(1-\left(1-\frac{1}{k}\right)^k\right)\cdot\varepsilon\cdot OPT\\
&\leq \frac{1}{e}\cdot OPT - \left(1-\frac{1}{e}\right)\cdot\varepsilon\cdot OPT.
\end{aligned}$$

Therefore,

$$\begin{aligned}
I(S_k) &= OPT - x_k\\
&\leq (1-1/e)\cdot(1-\varepsilon)\cdot OPT\\
&\leq (1-1/e-\varepsilon)\cdot OPT.
\end{aligned}$$

Thus, the lemma is proved. □