A

Social Influence Spectrum at Scale: Near-optimal Solutions for Multiple Budgets at Once

HUNG T. NGUYEN, Virginia Commonwealth University
PREETAM GHOSH, Virginia Commonwealth University
MICHAEL L. MAYO, US Army Engineer Research and Development Center
THANG N. DINH, Virginia Commonwealth University

Given a social network, the Influence Maximization (InfMax) problem seeks a seed set of k people that maximizes the expected influence for a viral marketing campaign. However, a solution for a particular seed size k is often not enough to make an informed choice regarding budget and cost-effectiveness.

In this paper, we propose the computation of *Influence Spectrum* (InfSpec), the maximum influence at **each** possible seed set size k within a given range $[k_{lower}, k_{upper}]$, thus provide optimal decision making for any availability of budget or influence requirements. As none of the existing methods for InfMax are efficient enough for the task in large networks, we propose LISA, an efficient approximation algorithm for InfSpec (and also InfMax) with the best known worst-case guarantees for billion-scale networks. LISA returns an $(1 - 1/e - \epsilon)$-approximate influence spectrum with high probability $(1 - \delta)$ where ϵ, δ are precision parameters provided by users. Using statistical decision theory, LISA has an asymptotic optimal running time (in addition to optimal approximation guarantee). In practice, LISA surpasses the state-of-the-art InfMax methods, taking less than 15 minutes to process a network of 41.7 million nodes and 1.5 billions edges.

CCS Concepts: •**Mathematics of computing** → **Approximation algorithms;** •**Networks** → **Online social networks;**

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Influence Maximization, Influence Spectrum, Approximation Algorithms

ACM Reference Format:
Hung T. Nguyen, Preetam Ghosh, Michael Mayo, Thang N. Dinh, 2016. Social Influence Spectrum at Scale: Near-optimal Solutions for Multiple Budgets at Once. *ACM Trans. Inf. Syst.* V, N, Article A (January YYYY), 25 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The Influence Maximization (InfMax) problem seeks to find a seed set of k *influential* individuals in a social network that can (directly and indirectly) influence the maximum number of people. It stands a fundamental problem in computational social networks with many applications in viral marketing, controlling epidemic disease, virus/worm propagation, and so on. Kempe et al. [Kempe et al. 2003] was the first to formulate InfMax as a combinatorial optimization problem on the two pioneering diffusion models, namely, independent Cascade (IC) and Linear Threshold (LT). Since InfMax is NP-hard,

---

Preliminary version of this work appeared in International Conference on Computational Social Networks, 2015 [Dinh et al. 2015].
Author's addresses: Hung T. Nguyen, Preetam Ghosh, Thang N. Dinh, Computer Science Department, Virginia Commonwealth University, Richmond, Virginia, 23284 and Michael L. Mayo, US Army Engineer Research and Development Center, 3909 Halls Ferry Road, Mississippi, 39180.

DOI: 0000001.0000001

they provide a natural greedy algorithm that yields $(1-1/e-\epsilon)$-approximate solutions for any $\epsilon > 0$. This celebrated work has motivated a vast amount of work on InfMax in the past decade [Leskovec et al. 2007; Chen et al. 2010; Goyal et al. 2011b; 2011a; Cohen et al. 2014; Ohsaka et al. 2014; Tang et al. 2014; Tang et al. 2015]. Nevertheless, the proposed methods cannot find satisfactory solutions in billion-scale networks. They either scale poorly [Leskovec et al. 2007; Goyal et al. 2011b; 2011a] or have no approximation guarantee [Chen et al. 2010]. Even the state-of-the-art method in [Tang et al. 2015] scales poorly with large numbers of seed nodes.

On top of the challenges in solving the InfMax in huge networks, we often need to find seed sets for multiple sizes k to make informed choices regarding budget and cost-effectiveness. For example, a viral campaign marketing might go through multiple phases. The planning of the expenses for each phase cannot be done properly without knowing the influence for multiple ranges of the number of seeds. Towards finding the optimal choices for multiple budgets, the authors in [Leskovec et al. 2007] optimize the size-influence ratio (the expected number of influenced individuals per seed node) or finding the min-seed set that can influence a large fraction of networks [Long and Wong 2011]. However, these approaches still give only one solution, that may not suite the multi-objective nature of decision making processes.

In this paper, we propose the computation of *Influence Spectrum* (InfSpec), the maximum influences (and the corresponding seed sets) for all possible seed sizes from $k = k_{lower}$ up to k_{upper}. The influence spectrum gives better insights for decision making and resource planning in viral marketing campaigns. Given the influence spectrum, we can find the solutions for not only InfMax but also cost-effective seed set [Leskovec et al. 2007] and min-seed set selection [Long and Wong 2011] problems (with the best approximation guarantees). Despite of many applictions of InfSpec, no one has ever considered computing InfSpec due to the perception that it seems extremely computationally expensive. The fact is computing InfSpec implies solving of $(k_{upper} - k_{lower} + 1)$ InfMax instances with seed size as large as n. Unfortunately, existing InfMax methods either do not scale well with large seed sets [Kempe et al. 2003; Leskovec et al. 2007; Goyal et al. 2011b; 2011a] or resort to heuristics [Chen et al. 2010], i.e., obtained results that could be arbitrarily worse than the optimal ones. Repeatedly running these algorithms on all different sizes $k \in [k_{lower}, k_{upper}]$ results in an unbearable running time for large ranges of $[k_{lower}, k_{upper}]$ even with the state-of-the-art for InfMax.

One might look into adapting existing InfMax methods for InfSpec task, however, doing so is difficult due to the following reasons:

— For the basic greedy approach, e.g. the original greedy algorithm [Kempe et al. 2003], CELF [Leskovec et al. 2007], CELF++ [Goyal et al. 2011a], one can run these algorithms with parameter k_{upper} and higher precision. However, this approach suffers from severe scalability issue for InfMax and thus, also for InfSpec.
— While recent state-of-the-art for InfMax [Tang et al. 2014; Tang et al. 2015; Nguyen et al. 2016] are scalable to very large networks, adapting them for the task faces non-trivial challenges. All TIM/TIM+ [Tang et al. 2014] and IMM [Tang et al. 2015] rely on generating approximately θ_k samples, where θ_k is a function of ϵ, δ, n and the unknown $OPT^{(k)}$, the maximum influence for any seed sets of size k. The most challenging part is to find a good lower-bound of $OPT^{(k)}$ and the better the lower-bound the less the number of samples. For example, by having a better bound on $OPT^{(k)}$, IMM is up to 100 times faster than TIM/TIM+. As θ_k is *not* a monotone function of k and estimating $OPT^{(k)}$ for various $k \in [k_{lower}, k_{upper}]$ is challenging, extending the theoretical approaches in [Tang et al. 2014] and [Tang et al. 2015] for InfSpec is not straightforward. The latest approaches in [Nguyen et al. 2016], named

SSA/D-SSA, attempt to use *less than* θ_k for InfMax by using out-of-sample validation to prove the optimality of the greedy solution over the generated samples. As the out-of-sample validation is designed to give just enough optimality evidence for a *particular* value of k, extending the approaches for InfSpec will increase the time to proving the optimality by a factor $k_{upper}-k_{lower}+1$, making the approach impractical.

We introduce LISA, an efficient approximation algorithm to compute InfSpec in billion-size networks. Given arbitrarily small $\epsilon, \delta > 0$, our algorithm has an expected running time $O((m+n)(k^* \log(n) + \log(k_{upper} - k_{lower} + 1))\epsilon^{-2})$[1] and output $(1 - \frac{1}{e} - \epsilon)$-approximate InfSpec with probability of $(1 - \delta)$. Also, LISA requires only an additional near linear space in addition to the graph. Compared with the recent approaches for InfMax, i.e. TIM/TIM+, IMM and SSA/D-SSA, LISA encloses a simple checking condition with a precomputed threshold Λ_k as opposed to unknown OPT_k or stopping point needed to be estimated (TIM/TIM+, IMM) or detected (SSA/D-SSA). The proposed algorithm has the best theoretical guarantees and outperforms the adaptations of the state-of-the-art methods for InfMax in practice. In particular, when $\epsilon = 0.1$ and $\delta = 1/n$, it takes about 15 minutes on a network with 41.7 million nodes and 1.5 billion edges under the LT model to find all seed sets of sizes from 1 to 1000. In comparison, it is up to 100 times faster than SSA/D-SSA [Nguyen et al. 2016] and IMM [Tang et al. 2015], the fastest known method with approximation guarantee for InfMax, when repeatedly running for different k in the range of 100, e.g. $k_{lower} = 1, k_{upper} = 100$. Compared with TIM and TIM+ [Tang et al. 2014], LISA is several magnitudes of order faster while providing guarantees for the whole spectrum of seed set sizes. Our contributions are summarized as follows.

— We introduce the problem of computing InfSpec that enables full spectrum analysis of influence in networks and provide efficient method to estimate influence spectrum with bounded error.
— We propose LISA an efficient approximation algorithm for InfSpec with a factor $(1 - 1/e - \epsilon)$. LISA is also capable of solving InfMax and Min-Seed selection problems (with best possible approximation factors). Moreover, it outputs InfMax solutions for all seed set sizes at the same time, effectively "*killing all birds with one stone*".
— We provide theoretical analysis to show the superiority of LISA in terms of time-complexity over existing methods (with approximation guarantees) for InfMax. Using sequential analysis theory, we derive the optimal number of samples needed to approximate influence of a queried seed set with a given level of accuracy.
— We perform experiments on large networks up to billions of edges. LISA is *easy to implement* and runs in *orders of magnitude faster* than the state-of-the-art InfMax methods adapted for InfSpec. Also, its *solution quality* is very close to that of the (slow) greedy method and outperforms IMM and TIM+'s for large-scale networks. Finally, in moderate and large networks, LISA uses the *least amount of memory* among all the methods.

Related works. Kempe et al. [Kempe et al. 2003] formulated the influence maximization problem as an optimization problem. They show the problem to be NP-complete and devise an $(1 - 1/e - \epsilon)$ approximation algorithm. Since InfMax encodes Max-Coverage problem as a special case, InfMax cannot be approximated within a factor $(1 - \frac{1}{e} + \epsilon)$ [Feige 1998] under a typical complexity assumption. Later, computing the exact influence is shown to be #P-hard [Chen et al. 2010]. Leskovec et al. [Leskovec et al. 2007] study the influence propagation in a different perspective in which they aim to find a set of nodes in networks to detect the spread of virus as soon as possible. They

[1] k^* is the seed size that results in the longest running time among $k_{lower} \leq k \leq k_{upper}$

improve the simple greedy method with the lazy-forward heuristic (CELF), which is originally proposed to optimize submodular functions in [Minoux 1978], obtaining an (up to) 700-fold speedup.

Several heuristics are developed to derive solutions in large networks. While those heuristics are often faster in practice, they fail to retain the $(1-1/e-\epsilon)$-approximation guarantee and produce lower quality seed sets. Chen et al. [Chen 2009] obtain a speed up by using an influence estimation for the IC model. For the LT model, Chen et al. [Chen et al. 2010] propose to use local directed acyclic graphs (LDAG) to approximate the influence regions of nodes. In a complement direction, there are recent works on learning the parameters of influence propagation models [Goyal et al. 2010; Kutzkov et al. 2013]. The influence maximization is also studied in other diffusion models including the majority threshold model [Dinh et al. 2014] or when both positive and negative influence are considered [Zhang et al. 2013; Li et al. 2013] and the propagation terminates after a fixed time [Dinh et al. 2014; Chen et al. 2012]. Recently, InfMax across multiple OSNs are studied in [Shen et al. 2012; Nguyen et al. 2013]. There are also applications of social influence in clustering methods [Zhou and Liu 2015] or in detecting misinformation in social networks [Huiling et al. 2016].

Recently, Borgs et al. [Borgs et al. 2014] make a theoretical breakthrough and present an $O(kl^2(m+n)\log^2 n/\epsilon^3)$ time algorithm for InfMax under IC model. Their algorithm (RIS) returns a $(1-1/e-\epsilon)$-approximate solution with probability at least $1-n^{-l}$. In practice, the proposed algorithm is, however, less than satisfactory due to the rather large hidden constants. In a sequential work, Tang et al. [Tang et al. 2014; Tang et al. 2015] reduce the running time to $O((k+l)(m+n)\log n/\epsilon^2)$ and show that their algorithm is also very efficient in large networks with billions of edges. Nevertheless, Tang's algorithm scales poorly with the number of seeds k and the estimation of the number of sampling times is both complicated and far from optimal. Recently, Nguyen et al. [Nguyen et al. 2016] propose the Stop-and-Stare framework which combines an independent check procedure to recompute the influence of candidate seed sets with sampling technique from [Borgs et al. 2014]. However, the algorithms in [Nguyen et al. 2016] contain a complicated parameter optimization and render difficulties in applying for InfSpec.

Influence Maximization on continuous-time diffusion network have recently attracted much attention on the line of works [Rodriguez and Schölkopf 2012; Du et al. 2013; Gomez-Rodriguez et al. 2016]. However, as pointed out in [Tang et al. 2015], the state-of-the-art methods [Du et al. 2013; Gomez-Rodriguez et al. 2016] on this diffusion model are significantly slower than those for the discrete ones, e.g., Independent Cascade, Linear Threshold. That is the complexity of methods in [Du et al. 2013; Gomez-Rodriguez et al. 2016] are larger than that of IMM [Tang et al. 2015] by a factor of at least k^2 where k is the size of the selected seed set.

Learning propagation model parameters also stands a critical position as the first starting point of our problem. Thus, it has received a good amount of interest in multiple places [Cha et al. 2009; Tang et al. 2009; Goyal et al. 2010; Liu et al. 2010; Saito et al. 2008]. [Cha et al. 2009; Tang et al. 2009; Goyal et al. 2010] focus on designing data mining or machine learning algorithms to extract influence cascade model parameters, e.g., fitting a variety of probabilistic models, from real datasets, e.g., action logs, connection networks. Based on link information and textual content associated with each node, [Liu et al. 2010] mines the direct and indirect topic-level influence by building a generative graphical model. Taking EM approach, Saito et al. [Saito et al. 2008] predict the diffusion probabilities by maximizing the likelihood of information diffusion episodes where an episode means a sequence of newly active nodes.

The rest of the paper is organized as follows: Section 2 presents the diffusion models considered in this work and states the InfSpec problem. Section 3 introduces the RIS sampling approach and proposes our sampling procedure for LT model as well as an efficient algorithm for computing InfSpec given a fixed seed set. That is followed by our proposed approximation algorithm LISA for InfSpec task in Section 4. Finally, we experimentally study the performance of LISA against other state-of-the-art methods in Section 5 and draw some conclusion in Section 6.

2. MODEL AND PROBLEM DEFINITION

In this section, we formally define the InfSpec problem and present an overview of Borgs et al. and Tang et al.'s methods [Borgs et al. 2014; Tang et al. 2014; Tang et al. 2015]. For simplicity, we focus on Linear Threshold (LT) model, however, our solution can be extended easily to Independent cascade (IC) model (Subsection 4.3).

2.1. Problem Definition

We abstract a social network using a weighted graph $\mathcal{G} = (V, E, w)$ with $|V| = n$ nodes and $|E| = m$ directed edges. Each edge $(u, v) \in E$ is associated with a weight $w(u, v) \in [0, 1]$ and $\sum_{u \in V} w(u, v) \leq 1$.

Linear Threshold (LT) model. Given a seed set $S \subseteq V$, the influence cascades in $\mathcal{G}$ happen in rounds. At round 0, all nodes in S are activated and the others are not activated. Each node v selects a random threshold λ_v uniformly at random in range $[0, 1]$. In a round $t \geq 1$, an inactivated node v becomes activated if $\sum_{\text{activated neighbor } u} w(u, v) \geq \lambda_v$. Once node v gets activated, it will remain activated til the end. The process stops when no more nodes get activated. Let $\mathbb{I}(S)$ denote the expected number of activated nodes given the seed set S, where the expectation is taken over all λ_v values from their uniform distribution. We call $\mathbb{I}(S)$ the *influence spread* of S under the LT model.

The LT model is equivalent to the reachability in the *live-edge* graphs, defined in [Kempe et al. 2003]: Given a graph $\mathcal{G} = (V, E, w)$, for every $v \in V$, select at most one of its incoming edges at random, such that the edge (u, v) is selected with probability $w(u, v)$, and no edge is selected with probability $1 - \sum_u w(u, v)$. Each live-graph G generated from $\mathcal{G}$ is also called a sample graph. The influence spread of a seed set S is same as the expected number of nodes reachable from S over all sample graphs.

Definition 2.1 (*Influence Maximization (InfMax)*). Given $k \leq n$, find a seed set of size k that maximizes $\mathbb{I}(S^{(k)})$ where $S^{(k)} \subseteq V$ and $|S^{(k)}| = k$.

Definition 2.2 (*Influence Spectrum (InfSpec)*). Given two integers k_{lower} and k_{upper}, for every $k \in [k_{lower}, k_{upper}]$, find a set of size k that maximizes $\mathbb{I}(S^{(k)})$ where $S^{(k)} \subseteq V$ and $|S^{(k)}| = k$.

When the context is clear, we also use InfSpec to indicate the influence values $\big(\mathbb{I}(S^{(k_{lower})}), \ldots, \mathbb{I}(S^{(k_{upper})})\big)$.

Complexity and Hardness. For each value of $k \in [k_{lower}, k_{upper}]$, we have an InfMax problem that finds the k-size seed set $S^{(k)}$ of maximum influence. Thus, InfSpec is at least as hard as InfMax. Since InfMax is an NP-hard problem, it follows that InfSpec is also an NP-hard problem. More than that, we cannot approximate InfMax with a factor $(1 - 1/e + \epsilon)$ unless $NP \subseteq DTIME(n^{\log \log n})$ [Feige 1998], we also obtain the same result for InfSpec problem.

Greedy algorithm for InfMax. The Greedy approach in [Kempe et al. 2003], referred to as the Greedy, starts with an empty seed set $S = \emptyset$, and iteratively adds to S a node

u that leads to the largest increase in the objective, i.e.,

$$u = \arg\max_{v \notin S}(\mathbb{I}(S \cup \{v\}) - \mathbb{I}(S))$$

To estimate $\mathbb{I}(S)$, we first generate a sample graph G of $\mathcal{G}$ using the live-edge model: select for each node $v \in \mathcal{G}$ at most one of its incoming edges at random, such that the edge (u, v) is selected with probability $w(u, v)$, and no edge is selected with probability $1 - \sum_u w(u, v)$. We then measure the number of nodes reachable from S in G, say $R_G(S)$. After generating "enough" sample graphs G (typically $n_s = 10,000$ samples [Kempe et al. 2003]), we can take the average of $R_G(S)$ as an estimation of $\mathbb{I}(S)$.

To select a node u, we may have to perform up to n estimations of $\mathbb{I}(.)$ that require generating n_s samples each. Thus, Greedy with its $O(k \times n_s \times mn)$ time complexity is computationally prohibitive for networks with millions of nodes. Later the heuristics CELF and CELF++ [Goyal et al. 2011a] are proposed to scale up the computation. Nevertheless, the greedy approach do not scale well for large networks.

2.2. InfSpec through InfMax approaches

We review the techniques with approximation guarantees for InfMax problem and discuss why they are either not scalable for large networks or not easily amenable for the task of InfSpec. We only focus on the rigorous methods that provide $(1 - 1/e - \epsilon)$ approximation guarantee.

Is InfSpec equivalent to InfMax when $k = k_{upper}$? It is tempting to solve InfSpec through solving InfMax with $k = k_{upper}$ to find a greedy solution $S^{(k_{upper})} = \{u_1, \ldots, u_{k_{upper}}\}$ and return for each $k \in [k_{lower}, k_{upper}]$, the k-prefix set of $S^{(k_{upper})}$ as solution of size k. We can use the basic greedy approach, such as the original greedy algorithm [Kempe et al. 2003], CELF [Leskovec et al. 2007] or CELF++ [Goyal et al. 2011a], with higher probability guarantees, however, the algorithms following this approach are not scalable for InfMax and thus, as well as for the InfSpec problem. They take days for solving InfMax on relatively small networks without few thousand nodes.

The above naive approach does not work with the recent state-of-the-art methods, namely TIM/TIM+ [Tang et al. 2014], IMM [Tang et al. 2015], SSA/D-SSA [Nguyen et al. 2016] either. These algorithms are based on *Reverse Influence Sampling* (RIS) proposed in [Borgs et al. 2014]. As shown in [Tang et al. 2014], there exists a theoretical threshold of θ_k for the number of required RIS samples. The solution found by the Max-Coverage greedy algorithm [Nemhauser et al. 1978] on θ_k RIS samples is guaranteed to be a $(1-1/e-\epsilon)$ approximation with high probability (whp). However, this threshold θ_k is a *non-monotone function of* k. The threshold θ_k in [Tang et al. 2014] is,

$$\theta_k = O(n\frac{\log(2/\delta) + \log\binom{n}{k}}{OPT^{(k)}\epsilon^2}). \quad (1)$$

Note that θ_k depends on both $\log\binom{n}{k}$, an increasing function of k, and the inverse of the optimal influence $OPT^{(k)}$ of size k which is a decreasing function of k.

Thus, the threshold $\theta_{k_{upper}}$ can be less or greater than θ_k for $k < k_{upper}$. It follows that the above approach does not provide the same $(1 - 1/e - \epsilon)$ guarantee for the seed set sizes $k < k_{upper}$.

Extending state-of-the art InfMax algorithms for InfSpec. We identify challenges in extending the best InfMax methods to solve InfSpec.

— TIM/TIM+ [Tang et al. 2014] shares a common framework with IMM [Tang et al. 2015] of estimating the influence of the optimal solution at the first phase and then using that number to estimate the RIS threshold θ_k. However, TIM/TIM+ [Tang et al. 2014] provide no bound on how close their estimation of the optimal influence is from

the true value and thus, the number of RIS samples can be arbitrarily larger than the threshold θ_k. In fact, as shown in IMM [Tang et al. 2015], with a better estimation of the optimal influence, IMM outperforms TIM/TIM+ on InfMax problem by a factor of 100x in terms of running time. On InfSpec problem with higher requirement of approximation guarantee on each seed set size, TIM/TIM+ need to estimate the optimal influences for different seed set sizes at once where each of them is not well estimated. Thus, these two algorithms will not perform well on large-scale networks.

— IMM [Tang et al. 2015] does provide a bound on the optimality estimation. However, IMM contains a parameter optimization step: for a size k, it tries to minimize the necessary number of samples θ_1 and θ_2 which depends on three other parameters $\epsilon_1 \leq \epsilon, \delta_1, \delta_2$ where $\delta_1 + \delta_2 = \delta$ as follows:

$$\theta_1 = \frac{2n \log(1/\delta_1)}{OPT^{(k)} \epsilon_1^2}; \theta_2 = \frac{(2 - 2/e) n \log(\binom{n}{k}/\delta_2)}{OPT^{(k)} (\epsilon - (1 - 1/e)\epsilon_1)^2}. \quad (2)$$

Among θ_1 and θ_2, whichever larger is the required RIS samples. However, generalizing this optimization problem for all $k \in [k_{lower}, k_{upper}]$ to address InfSpec problem results in a huge number of parameters which all depend on each other to optimize. Thus, generalizing IMM for InfSpec is not trivial.

— SSA/D-SSA [Nguyen et al. 2016] takes a very different approach to achieve another theoretical RIS threshold which is at most θ_k for a single size k. It relies on a separate checking procedure to recompute the influence of the candidate seed set and detect the first time at which the solution meets the requirement. Thus, SSA/D-SSA may even use fewer samples than the threshold θ_k. However, SSA/D-SSA have a complicated precision parameter optimization, i.e. $\epsilon_1, \epsilon_2, \epsilon_3, \delta_1, \delta_2$ satisfying $\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2 + \epsilon_3(1 - 1/e) \leq \epsilon$ and $\delta_1 + \delta_2 \leq \delta$ for a value of size k to achieve the optimal sampling requirement. For a range of sizes $k \in [k_{lower}, k_{upper}]$, the number of precision parameters are exploded and the optimization over these parameters is very difficult. This may require a thorough investigation of the SSA/D-SSA and major modifications are unavoidable.

In contrast to the existing algorithms for InfMax, the algorithmic design of LISA is sharply distinctive. LISA provides an InfSpec approximation guarantee at *every iteration* and uses a simple stopping condition to recognize the point at which the provided guarantee meets the requirement of (ϵ, δ). The stopping condition compares the coverage of the seed sets on generated RIS samples with a precomputed threshold. The check is easily done and fast. Thus, LISA neither needs to estimate the optimal influences for all seed sizes as TIM/TIM+ and IMM nor rely on an independent checking procedure to recompute the influences of the candidate seed sets as SSA/D-SSA.

3. SIMULTANEOUS HIGH-CONFIDENT ESTIMATION OF INFLUENCE SPECTRUM

In this section, we investigate the problem of obtaining high-confident and *bounded-error* estimation of influence of *multiple* seed sets simultaneously. This is critical for knowing whether or not we have sufficient samples to provide the guarantees on the solution of InfSpec. Given a node order $S = \{v_1, \ldots, v_n\}$, e.g., like the one obtained through the greedy approach, we wish to compute the influence of all k-prefixes with $k_{lower} \leq k \leq k_{upper}$, i.e., to calculate InfSpec $\mathbb{I}(S^{(k)}), \forall k \in [k_{lower}, k_{upper}]$ where $S^{(k)} = \{v_1, \ldots, v_k\}$. Computing exact InfSpec is intractable as computing the influence of a single seed set is already #P-hard [Kempe et al. 2003].

Even approximating these influence values with an ϵ-error is difficult with existing methods. Previous works [Kempe et al. 2003; Chen et al. 2010] have to resort to estimation by simulating the influence cascades from the selected seeds many times and take the average of those simulated influence. This approach has the complexity

Table I: Table of Notations

Notation	Description
n, m	#nodes, #links of graph $\mathcal{G} = (V, E)$, respectively
k_{lower}, k_{upper}	Lower and upper numbers of selected seed nodes
$\mathbb{I}(S)$	Influence Spread of seed set $S \subseteq V$. For $v \in V$, $\mathbb{I}(v) = \mathbb{I}(\{v\})$
$OPT^{(k)}$	The maximum $\mathbb{I}(S^{(k)})$ for any size-k node set $S^{(k)}$
$S^{(k)*}$	An optimal size-k seed set, i.e., $\mathbb{I}(S^{(k)*}) = OPT^{(k)}$
$\hat{S}^{(k)}$	The returned seed set of size k, $\hat{S}^{(k)} = \{\hat{v}_1, \ldots, \hat{v}_k\}$
$m_{\mathcal{H}}$	#hyperedges in hypergraph $\mathcal{H}$
$deg_{\mathcal{H}}(S), S \subseteq V$	#hyperedges incident at some node in S. Also, $deg_{\mathcal{H}}(v)$ for $v \in V$
c	Sampling constant $c = 2(e-2) \approx \sqrt{2}$
$M_k, \mathcal{M}$	$M_k = \binom{n}{k}, \mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}} (M_k + 1)$
Υ	$\Upsilon = 8c(1 - \frac{1}{2e})^2(\log \frac{2}{\delta} + \log \mathcal{M} + \ln(\log_2(\frac{n}{k_{lower}})))\frac{1}{\epsilon^2}$ (Note $\log \mathcal{M} < k^* \log \frac{n(k_{upper}-k_{lower}+1)}{k^*}$ where $\binom{n}{k^*} \geq \binom{n}{k}, \forall k \in [k_{lower}, k_{upper}]$)
Λ	$\Lambda = (1 + \frac{e}{2(2e-1)}\epsilon)\Upsilon$

of $O((m+n)R)$ where R is the number of simulations. However, R is very large, i.e. $O(\epsilon^2 k^2 n \log(n^2 k))$ [Tang et al. 2014; Chen et al. 2013]. Recent methods [Borgs et al. 2014; Tang et al. 2014; Tang et al. 2015] use Reverse Influence Sampling (RIS) to give high-confidence estimations for the influence of a single seed set. However, these methods are still not scalable for large ranges of $k \in [k_{lower}, k_{upper}]$.

To this end, we propose an efficient algorithm to give high-confidence estimation for multiple seed sets at scale. In fact, the time needed to estimate for multiple seed sets is the same with the time to evaluate the influence for a single seed set.

Our algorithm is built on top of the reverse influence sampling technique [Borgs et al. 2014]. The RIS procedure to generate a random hyperedge $\mathcal{E}_j \subseteq V$ in LT model is summarized in Algorithm 1. After choosing a starting node u randomly, we attempt to select an *in-neighbor* v of u, i.e. (v, u) is an edge of $\mathcal{G}$, according to the edge weights. Then we "move" to v and repeat, i.e. to continue the process with v replaced by u. The procedure stops when we encounter a previously visited vertex or no edge is selected. The hyperedge is then returned as the set of nodes visited along the process.

ALGORITHM 1: RIS-LT: Reverse Influence Sampling in LT model

Input: Weighted graph $\mathcal{G} = (V, E, w)$
Output: A random hyperedge $\mathcal{E}_j \subseteq V$

```
1  E ← ∅
2  Pick a node v uniformly at random.
3  repeat
4  |  Add v to E_j
5  |  Attempt to select an edge (u, v) using live-edge model
6  |  if edge (u, v) is selected then
7  |  |  Set v ← u
8  |  end
9  until (v ∈ E_j) OR (no edge is selected);
10 Return E_j
```

The key insight into why random hyperedges generated via RIS can capture the influence landscape is stated in the following lemma.

LEMMA 3.1. *Given a fixed seed set $S \subseteq V$, for a random hyperedge $\mathcal{E}_j$,*

$$\Pr[\mathcal{E}_j \cap S \neq \emptyset] = \frac{\mathbb{I}(S)}{n}$$

The proof is similar to that for IC model in [Borgs et al. 2014] and is omitted.

Thus we can apply Monte-Carlo method to estimate the influence of a given seed set S, i.e., to generate enough hyperedges (aka samples) and compute the frequency that the hyperedges intersect with S. Even better, we only need to generate the hyperedges once, and can reuse the hyperedges to approximate the influence of as many seed sets as we want. This is a huge advantage comparing to the traditional Greedy [Kempe et al. 2003], in which we have to perform an excessive number of BFS to estimate nodes' influence. All we need to figure out is the number of sample times (i.e. number of hyperedges) needed to estimate nodes' influence at a desired level of accuracy.

3.1. Number of Samples (Hyperedges)

This section focuses on the number of samples (hyperedges) needed to achieve a pre-determined performance guarantee. As the number of samples directly decides the running time, it is critical to minimize the number of samples (preserving the same performance guarantees). For example, Borgs et al.'s method requires at least $48\frac{(m+n)\log n}{\epsilon^3 OPT^{(k)}}$ hyperedges to find a $(1-1/e-\epsilon)$-approximate of InfMax with probability at least $1-1/n^l$, while Tang et al.'s [Tang et al. 2014] needs only $(8+\epsilon)\frac{(k+l)(m+n)\log(n)}{\epsilon^2 OPT^{(k)}}$ hyperedges to provide the same guarantees. Here $OPT^{(k)} = \max_{|S|=k, S\subseteq V}\{\mathbb{I}(S)\}$, the maximum influence of any size-k seed set. Hence, the Tang et al.'s is asymptotically $\frac{1}{\epsilon}$ times faster than the Borgs et al.'s.

Let Z be a random variable distributed in $[0,1]$ with mean $\mathbb{E}[Z] = \mu$ and variance σ_Z^2. Let $Z_1, Z_2, \ldots, Z_T$ be independently and identically distributed (i.i.d.) realizations of Z. A *Monte Carlo estimator* of μ_Z is,

$$\tilde{\mu} = \frac{1}{T}\sum_{i=1}^{T} Z_i. \tag{3}$$

$\tilde{\mu}$ is said to be an (ϵ,δ)-approximation of μ, for $0 < \epsilon, \delta \leq 1$, if

$$\Pr[|\tilde{\mu} - \mu| \leq \epsilon\mu] \geq 1-\delta. \tag{4}$$

Let $\rho(\epsilon) = \max\{\sigma^2, \epsilon\mu\}$. The Generalized Zero-One Estimator Theorem in [Dagum et al. 2000] proves that if

$$T \geq 2c\ln\frac{2}{\delta}\frac{\rho(\epsilon)}{\epsilon^2\mu^2}, \tag{5}$$

where $c = 2(e-2)$, then $\tilde{\mu} = \frac{1}{T}\sum_{i=1}^{T} Z_i$ is an (ϵ,δ)-approximation of μ. Moreover, the number of sampling time is (asymptotically) optimal (by a constant factor) [Dagum et al. 2000]. Additionally from [Dagum et al. 2000], the second way of achieving an (ϵ,δ)-approximation of μ is based on the condition that,

$$\sum_{i=1}^{T} Z_i \geq 1 + (1+\epsilon)2c\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}, \tag{6}$$

and the necessary number of samples to achieve this condition is also asymptotically optimal.

In this paper, we are interested in the random variable Z with realizations

$$Z_j = \min\{|S \cap \mathcal{E}_j|, 1\}, \tag{7}$$

where S is a fixed seed set and $\mathcal{E}_j$ is a random hyperedge generated by Algorithm 1. From Lemma 3.1, Z is a random variable with mean $\mu_Z = \mathbb{I}(S)/n$ and variance $\sigma_Z^2 = (1 - \mu_Z)\mu_Z$.

A major obstacle in using Eq. (5) to derive the optimal number of samples is that we do not know σ_Z^2 and μ_Z, the quantity we are trying to estimate. Let $S^{(k)*} = \arg\max_{|S|=k, S\subseteq V}\{\mathbb{I}(S)\}$, and $OPT^{(k)} = \mathbb{I}(S^{(k)*})$. If we can come up with a close bound on $OPT^{(k)}$, we will know the necessary number of hyperedges to capture the influence landscape. After that, InfMax and InfSpec can be reduced to the classic Max-Coverage problem [Vazirani 2001] as shown in [Borgs et al. 2014; Tang et al. 2014].

Thus, the key to the efficiency of the previous studies in [Borgs et al. 2014; Tang et al. 2014; Tang et al. 2015] are the methods to probe and estimate the value of $OPT^{(k)}$. With the better probing and estimating techniques, TIM/TIM+ and IMM in [Tang et al. 2014; Tang et al. 2015] reduce the time-complexity in [Borgs et al. 2014] by a factor $O(1/\epsilon)$, making the first scalable method for InfMax in billion-size networks. However, the number of samples in [Tang et al. 2014; Tang et al. 2015] is still far from optimal, especially for large seed sets. As a consequence, the two algorithms scale poorly with large number of seeds.

3.2. Efficient Influence Spectrum Estimation

ALGORITHM 2: Efficient Influence Spectrum Validation Algorithm (EIVA)

Input: Weighted graph $\mathcal{G}$, a seed set $S = \{v_1, v_2, \ldots, v_n\}$, k_{lower}, k_{upper} that $1 \le k_{lower} \le k_{upper} \le n$ and $\epsilon, \delta \in (0, 1)$

Output: (ϵ, δ)-approximation of $\mathbb{I}(\hat{S}^{(k)}), \forall k \in [k_{lower}, k_{upper}]$

$\Lambda_L \leftarrow 1 + 2c(1+\epsilon)(\ln\frac{2}{\delta} + \ln(k_{upper} - k_{lower} + 1))\frac{1}{\epsilon^2}$
$T \leftarrow 0, deg_k \leftarrow 0, \forall k = 1, \ldots, n$
repeat
  Generate a random hyperedge $\mathcal{E}_j \leftarrow RIS - LT(\mathcal{G})$
  $t = \arg\min_i\{v_i \in \mathcal{E}_j\}$
  $deg_t \leftarrow deg_t + 1$
  $T \leftarrow T + 1$
until $\sum_{i=1}^{k_{lower}} deg_i \ge \Lambda_L$;
$\hat{\mathbb{I}}_{k_{lower}} = deg_{k_{lower}} \cdot n/T$
for $k = (k_{lower} + 1) : k_{upper}$ **do**
  $\hat{\mathbb{I}}_k \leftarrow \hat{\mathbb{I}}_{k-1} + deg_k \cdot n/T$
end
return $\hat{I} = \{\hat{\mathbb{I}}_k | k \in [k_{lower}, k_{upper}]\}$

In this section, we provide a fast and memory-efficient algorithm, called EIVA, to estimate the influence spectrum in arbitrarily good accuracy with high probability. Given an ordered set $S = \{v_1, \ldots, v_n\}$ and two integers $1 \le k_{lower} \le k_{upper} \le n$, we want to compute all $\mathbb{I}(S^{(k)}), \forall k \in [k_{lower}, k_{upper}]$ where $S^{(k)} = \{v_1, v_2, \ldots, v_k\}$. Here we assume $S = \{v_1, \ldots, v_n\}$ is fixed and given as a set of seed nodes which can be the output of an InfSpec algorithm. Denote $\hat{\mathbb{I}}(S^{(k)})$ the estimated value of $\mathbb{I}(S^{(k)})$ returned by EIVA algorithm. EIVA guarantees that $\hat{\mathbb{I}}(S^{(k)}), \forall k \in [k_{lower}, k_{upper}]$ are within $(1 \pm \epsilon)$

the actual values with probability at least $1-\delta$ where $\epsilon, \delta \in (0,1)$ are arbitrarily small values chosen by the users.

Specifically, EIVA, shown in Algorithm 2, repeatedly generates a hyperedge $\mathcal{E}_j$ in each step. It then looks for the smallest index t that $v_t \in \mathcal{E}_j$. Observe that all seed sets $S^{(k)}, k \geq t$ will cover hyperedge $\mathcal{E}_j$. Instead of increasing the value of all $deg_k, k \geq t$, EIVA only increases deg_k by one. Finally, the values of deg_k will be aggregated at the end, lines 10 and 11. This smart update strategy reduces the worst-case time-complexity per hyperedge from $O(n)$ to $O(1)$. Hence, we will be able to compute all the influence of the seed sets much faster.

LEMMA 3.2. *EIVA (Algorithm 2) computes (ϵ, δ)-approximate for the influences of all seed sets in time $O(\epsilon^{-2}(\ln \frac{2}{\delta} + \ln(k_{upper} - k_{lower} + 1))(m+n))$ and **only an** $\theta(\mathbf{n})$ **additional space** (excluding the space to store the graph).*

PROOF. The complexity analysis is similar to that of LISA algorithm which will be presented in Subsection 4.2. The space complexity is followed directly from the fact that EIVA does not store hyperedges but only need a single array to store the values of $deg_k, k = 1, \ldots, n$, thus its space-complexity is $\theta(n)$. Here we prove the approximation factor. First, due to the condition in line 8, we have,

$$\sum_{i=1}^{k_{upper}} deg_i \geq \sum_{i=1}^{k_{upper}-1} deg_i \geq \cdots \geq \sum_{i=1}^{k_{lower}} deg_i \geq \Lambda_L \tag{8}$$

Thus, for any $k \in [k_{lower}, k_{upper}]$, based also on the condition of achieving an (ϵ, δ)-approximation of $\mathbb{I}(S^{(k)})$ for a single set $S^{(k)}$ in Eq. 6, we obtain,

$$\Pr[|\hat{\mathbb{I}}(S^{(k)}) - \mathbb{I}(S^{(k)})| \leq \epsilon \mathbb{I}(S^{(k)})] \geq 1 - \frac{\delta}{k_{upper} - k_{lower} + 1} \tag{9}$$

Taking the union bound of the above inequality over all values of k from k_{lower} to k_{upper} (there are $k_{upper} - k_{lower} + 1$ such values) gives,

$$\Pr[|\hat{\mathbb{I}}(S^{(k)}) - \mathbb{I}(S^{(k)})| \leq \epsilon \mathbb{I}(S^{(k)}), \forall k = k_{lower}, \ldots, k_{upper}] \geq 1 - \delta \tag{10}$$

which proves the (ϵ, δ)-approximate for the influences of all seed sets S_k where $k \in [k_{lower}, k_{upper}]$ and completes the proof. □

Comparing with the complexity of naive algorithm for computing InfSpec by cascade simulation, we see that EIVA saves a factor of $R\frac{k_{upper}-k_{lower}+1}{\ln(k_{upper}-k_{lower}+1)}$. More importantly, EIVA guarantees an (ϵ, δ)-approximation for the returned InfSpec estimation.

4. LISA APPROXIMATION ALGORITHM FOR IDENTIFY MULTIPLE SEED SETS

In this section, we propose LISA approximation algorithm that returns a $(1 - 1/e - \epsilon)$-approximate InfSpec with probability at least $(1-\delta)$ for any constant $\epsilon, \delta \in (0,1)$.

Define $\Upsilon = 8c(1-\frac{1}{2e})^2(\log \frac{2}{\delta} + \log \mathcal{M} + \ln(\log_2(\frac{n}{k_{lower}})))\frac{1}{\epsilon^2}$ where $\mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}}(M_k+1)$ and $M_k = \binom{n}{k}$. Our algorithm, named LISA, is presented in Algorithm 3. A stream of hyperedges is generated by RIS-LT (Algorithm 1) and used in the iterations of the algorithm (Line 1). It consists of a main loop of iterations. At iteration $i = 1, 2, \ldots,$ LISA 1) fetches $\Upsilon 2^i$ hyperedges into the hypergraph and 2) solves an instance of Max-Coverage using a greedy approach (Algorithm 4). The algorithm terminates when the stopping condition in Line 8 is satisfied and returns the seed sets together with their approximated influences.

Borgs et al. [Borgs et al. 2014] generates hyperedges until a pre-defined number of edges explored by the algorithm and only provide a low successful probability $2/3$.

ALGORITHM 3: LISA Influence Spectrum Algorithm

Input: Precision $\epsilon \in (0,1)$, $\delta \in (0,1)$, weighted graph $\mathcal{G}$ and min/max seed set sizes k_{lower}, k_{upper}
Output: $\{\hat{S}^{(k)}, k \in [k_{lower}, k_{upper}]\}$ and their influences $\{\hat{\mathbb{I}}(\hat{S}^{(k)}), \in [k_{lower}, k_{upper}]\}$.

Generate a stream of hyperedges $\mathcal{E}_1, \mathcal{E}_2, \dots$ using RIS-LT in Alg. 1
Compute $\Lambda = (1 + \frac{e}{2(2e-1)}\epsilon)\Upsilon$
Initialize $i = 1$ and $\mathcal{H} \leftarrow (V, \mathcal{E} = \emptyset)$
repeat
  $\mathcal{H} \leftarrow (V, \mathcal{E} = \{\mathcal{E}_1, \dots, \mathcal{E}_{\Upsilon 2^i}\}); i = i + 1$
  $\hat{S} =$ Max-Coverage$(\mathcal{H}, k_{upper})$ `// Assume` $\hat{S} = \{\hat{v}_1, \hat{v}_2, \dots, \hat{v}_{k_{upper}}\}$
  $\hat{S}^{(k)} = \{\hat{v}_1, \dots, \hat{v}_k\}, \forall k \in [k_{lower}, k_{upper}]$
until $deg_{\mathcal{H}}(\hat{S}^{(k_{lower})}) \geq \Lambda$ **or** $i \geq \log_2(\frac{n}{k_{lower}})$;
Compute $\hat{\mathbb{I}}(\hat{S}^{(k)}) = \frac{deg_{\mathcal{H}}(\hat{S}^{(k)}) \cdot n}{m_{\mathcal{H}}}, \forall k \in [k_{lower}, k_{upper}]$
return $\{\hat{S}^{(k)}, k \in [k_{lower}, k_{upper}]\}$ and $\{\hat{\mathbb{I}}(\hat{S}^{(k)}), k \in [k_{lower}, k_{upper}]\}$

ALGORITHM 4: Max-Coverage

Input: Hypergraph $\mathcal{H}$ and maximum number of seeds k_{upper}.
Output: Seed set $\hat{S}$

$S = \emptyset$
for $i = 1 : k_{upper}$ **do**
  $\hat{v} \leftarrow \arg\max_{v \in V}(deg_{\mathcal{H}}(\hat{S} \cup \{v\}) - deg_{\mathcal{H}}(\hat{S}))$
  Add $\hat{v}$ to $\hat{S}$
end
return $\hat{S}$

While the authors suggest that their algorithm can be repeated multiple times to boost up the success probability, this approach leads to a very inefficient implementation. Tang et al. [Tang et al. 2014] estimate $OPT^{(k)}$ via the average cost of RIS, called EPT^k. However, their approach still requires generating as many as k times more hyperedges than necessary. Differently, we propose a novel stopping rule: *we stop generating hyperedges once the degrees of all seed sets in the hypergraph reach their corresponding constants* Λ. Here, the degree of a node in the hypergraph is the number of hyperedges that contain the node. Later we show our stopping rule guarantees a 'rich' enough hypergraph to estimate nodes' influence and small enough hyperedges to make an efficient algorithm.

Our algorithm is easy to implement and requires no parameters rather than ϵ and δ. In practice, it scales very well with billion-size networks and large seed sets. It proves to be the fastest algorithm known for InfSpec while maintaining superior solution quality at the same time.

4.1. Approximation Guarantees

In this subsection, we will prove the approximation factor of LISA to be $(1 - \frac{1}{e} - \epsilon)$. In our context of InfSpec, the $(1 - \frac{1}{e} - \epsilon)$-approximation guarantee means that for all sizes $k \in [k_{lower}, k_{upper}]$, $\mathbb{I}(\hat{S}^{(k)}) \geq (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{(k)*})$ where $\hat{S}^{(k)} = \{\hat{S}_1, \dots, \hat{S}_k\}$ and $S^{(k)*}$ is a fixed optimal seed set of size k with the optimal influence of $OPT^{(k)}$. We say that an InfSpec algorithm returns an $(1 - \frac{1}{e} - \epsilon)$-approximate solution $\hat{S}$ with probability at least $(1 - \delta)$ if,

$$\Pr\left[\mathbb{I}(\hat{S}^{(k)}) \geq (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{(k)*}), \forall k \in [k_{lower}, k_{upper}]\right] \geq 1 - \delta \tag{11}$$

In other words, it is equivalent to show that,

$$\Pr\left[\exists k \in [k_{lower}, k_{upper}], \mathbb{I}(\hat{S}^{(k)}) < (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{(k)*})\right] < \delta \tag{12}$$

The following will prove that LISA returns a $(1 - 1/e - \epsilon)$-approximate solution $\hat{S}$ with probability at least $(1 - \delta)$ where ϵ and δ are parameters in Alg. 3 (Theorem 4.4).

Roadmap. To prove the Eq. 12, we will intermediately prove a stronger inequality,

$$\sum_{k=k_{lower}}^{k_{upper}} \Pr[\mathbb{I}(\hat{S}^{(k)}) < (1 - \frac{1}{e} - \epsilon)\mathbb{I}(S^{(k)*})] < \delta \tag{13}$$

which implies Eq. 12 due to the inequality between probability of a union of events and sum of probabilities of individual events. More specifically, we will show that LISA's stopping condition (Line 8 of Alg. 3) guarantees the kth term in the sum to be bounded by $\delta \frac{M_k+1}{\mathcal{M}}$ where $M_k = \binom{n}{k}, \mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}}(M_k + 1)$ and thus, the Eq. 13 follows.

In order to prove the approximation guarantee of each seed set $\hat{S}^{(k)}$, we consider each iteration $i = 1, 2, \dots$ and prove that with the number of samples $N_i = \Upsilon 2^i$, any seed set $S^{(k)}$ of size k is approximated within an error bound with a high probability (Lemma 4.1). Based on that and the $(1 - \frac{1}{e})$-approximation guarantee of the Max-Coverage algorithm on finding candidate seed set $\hat{S}^{(k)}$, we obtain an approximation guarantee which depends on i in iteration i of $\hat{S}^{(k)}$ (Lemma 4.2). Finally, using the result on approximation guarantee at every iteration, we prove that at termination point (stopping condition met), $\hat{S}^{(k)}$ is within $(1 - \frac{1}{e} - \epsilon)$ with a high probability of $1 - \frac{(M_k+1)\delta}{\mathcal{M}}$ (Lemma 4.3) through union bound over all possible seed sets of size k and all iterations.

We first present the following results that will be used in our proofs. For a node set S and a hyperedge $\mathcal{E}_j$, recall the random variable Z_j defined in Eq. 7,

$$Z_j = \min\{|S \cap \mathcal{E}_j|, 1\}, \tag{14}$$

Thus, the series of hyperedges in $\mathcal{H}$ corresponds to a sequence of random variables of Z_j, denoted by $\{Z_1, Z_2, \dots\}$. Intuitively, since the hyperedges are generated independently, the resulted sequence of $Z_1, Z_2, \dots$ should also be independent and identically distributed in $[0, 1]$. However, similar to the Stopping Rule Algorithm in [Dagum et al. 2000] that LISA creates a dependency on the samples by stopping the algorithm when some condition is satisfied. LISA jumps to the next round when $deg_{\mathcal{H}}(\hat{S}^{(k_{lower})}) \geq \Lambda$ or, $\sum_{i=1}^{|\mathcal{H}|} Z_i \geq \Lambda$ where Z_j corresponds to $\hat{S}^{(k_{lower})}$, is not met and hence, whether we generate more samples depending on the current set of hyperedges. Interestingly, similar to the case of Stopping Rule Algorithm in [Dagum et al. 2000], the sequence $\{Z_1, Z_2, \dots\}$ is related to a *martingle* defined over the random variables

$$X_i = \sum_{j=1}^{i}(Z_j - \mu_Z). \tag{15}$$

Based on martingale theory, we obtain the following results.

Let $Z_1, Z_2, ...$ be the stream of random variables according to Z random variable in the interval $[0, 1]$ with mean μ_Z and variance σ_Z^2 and $\hat{\mu}_Z = \frac{1}{T}\sum_{i=1}^{T} Z_i$ be an estimate

of μ_Z, for any fixed $T > 0, 0 \geq \epsilon \geq 1$. We have the following two Chernoff-like bounds [Dinh et al. 2015]:

$$\Pr[\hat{\mu}_Z \geq (1+\epsilon)\mu_Z] \leq e^{\frac{-T\mu_Z\epsilon^2}{2c}} \tag{16}$$

and,

$$\Pr[\hat{\mu}_Z \leq (1-\epsilon)\mu_Z] \leq e^{\frac{-T\mu_Z\epsilon^2}{2c}}. \tag{17}$$

where $c = 2(e-2)$ and e is the base of the natural logarithm.

For convenience, denote $\mu_k = \frac{\mathbb{I}(S^{(k)})}{n}, \hat{\mu}_k = \frac{\hat{\mathbb{I}}(S^{(k)})}{n}$ and $\mu_k^* = \frac{OPT^{(k)}}{n}, \hat{\mu}_k^* = \frac{\hat{\mathbb{I}}(S^{(k)*})}{n}$. Recall that in each iteration of the repeat loop in LISA, it fetches $\Upsilon 2^i$ random variables Z_j into the hypergraphs for use. Using the above two probabilistic inequalities, we obtain the following results for iteration i:

LEMMA 4.1. *At the iteration i of LISA, for a size $k \in [k_{lower}, k_{upper}]$ and any set $S_k \subseteq V$ of size k, the following two inequalities hold,*

$$\Pr[\hat{\mathbb{I}}(S^{(k)}) \geq \mathbb{I}(S^{(k)}) + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}] \leq \frac{\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{18}$$

and

$$\Pr[\hat{\mathbb{I}}(S^{(k)}) \leq \mathbb{I}(S^{(k)}) - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}] \leq \frac{\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{19}$$

where $\mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}} M_k$ and $M_k = \binom{n}{k} + 1$.

PROOF. The proof follows directly from Eq. 16 and Eq. 17 with the number of random variables being $N_i = \Upsilon 2^i$. The left hand size of Eq 18 is rewritten as follows,

$$\Pr[\hat{\mu}_k \geq \mu_k + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}\mu_k^*] = \Pr[\hat{\mu}_k \geq \mu_k + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}\frac{\mu_k^*}{\mu_k}\mu_k] \tag{20}$$

where $\hat{\mu}_k$ is an approximate of μ_k using $N_i = \Upsilon 2^i$ samples.

Thus, applying the bound on Eq. 16 on the above probability, we obtain,

$$\Pr[\hat{\mu}_k \geq \mu_k + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}\frac{\mu_k^*}{\mu_k}\mu_k] \leq e^{-\frac{T\mu_k e^2\epsilon^2\mu_k^{*2}}{2c(2e-1)^2 2^i\mu_k^*\mu_k^2}} \tag{21}$$

$$\leq e^{-\frac{8c(1-\frac{1}{2e})^2(\log\frac{2}{\delta}+\log\mathcal{M}+\ln(\log_2(\frac{n}{k_{lower}})))\frac{1}{\epsilon^2}2^i e^2\epsilon^2}{2c(2e-1)^2 2^i}\frac{\mu_k^*}{\mu_k}} \leq \frac{\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{22}$$

where $N_i = \Upsilon 2^i \geq \Upsilon 2^i$ and $\frac{\mu_k^*}{\mu_k} \geq 1$.

Similarly, we can prove Eq. 19 using the bound in Eq. 17. Thus, we complete the proof of Lemma 4.1 □

Using the results of Lemma 4.1, we prove the following result regarding the approximation guarantee of the candidate solution $\hat{S}^{(k)}$ returned by Max-Coverage algorithm for size k in iteration i.

LEMMA 4.2. *At the iteration i of LISA, for a size $k \in [k_{lower}, k_{upper}]$, the candidate solution $\hat{S}^{(k)}$ returned by Max-Coverage satisfies the following,*

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \frac{\epsilon}{(2^i\mu_k^*)^{1/2}})OPT^{(k)}] \leq \frac{(M_k+1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{23}$$

PROOF. Since Lemma 4.1 holds for any subset $S^{(k)} \subseteq V$ of size k (there are $M_k = \binom{n}{k}$ of such sets), from Eq. 18 of Lemma 4.1 and apply union bound over all subsets $S^{(k)} \subseteq V$ of size k, we obtain that,

$$\Pr[\exists S^{(k)} \subseteq V, |S^{(k)}| = k, \hat{\mathbb{I}}(S^{(k)}) \geq \mathbb{I}(S^{(k)}) + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}] \leq \frac{M_k\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{24}$$

In other words, the above inequality provides an upper-bound on the approximation quality of $\hat{\mathbb{I}}(S^{(k)})$ compared to the true values over all possible subsets of size k.

Moreover, the candidate solution $\hat{S}^{(k)}$ returned by Max-Coverage algorithm is one of those M_k subsets of size k. Thus, Eq. 24 infers,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \geq \mathbb{I}(\hat{S}^{(k)}) + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}] \leq \frac{M_k\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{25}$$

Apply the inequality in Eq. 19 on the optimal solution $S^{(k)*}$ for size k, we also have,

$$\Pr[\hat{\mathbb{I}}(S^{(k)*}) \leq OPT^{(k)} - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}] \leq \frac{\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})} \tag{26}$$

Combining Eq. 25 and Eq. 26 gives the following statement: either $\hat{\mathbb{I}}(\hat{S}^{(k)}) \geq \mathbb{I}(\hat{S}^{(k)}) + \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}$ or $\hat{\mathbb{I}}(S^{(k)*}) \leq OPT^{(k)} - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}$ happen inclusively with a probability of at most $\frac{(M_k+1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}$. That implies that both

$$\mathbb{I}(\hat{S}^{(k)}) \geq \hat{\mathbb{I}}(\hat{S}^{(k)}) - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)} \tag{27}$$

and

$$\hat{\mathbb{I}}(S^{(k)*}) \geq OPT^{(k)} - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)} \tag{28}$$

happen with a probability of at least $1 - \frac{(M_k+1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}$.

Furthermore, since $\hat{S}^{(k)}$ is obtained from the greedy Max-Coverage algorithm, [Nemhauser et al. 1978] gives the following result,

$$\hat{\mathbb{I}}(\hat{S}^{(k)}) \geq (1-\frac{1}{e})\hat{\mathbb{I}}(\hat{S}^{(k)}_{max}) \geq (1-\frac{1}{e})\hat{\mathbb{I}}(S^{(k)*}), \tag{29}$$

where $\hat{S}^{(k)}_{max}$ is the optimal solution of the coverage problem of selecting k nodes covering the maximum number of hyperedges. Incorporating this result with Eq. 27 and Eq. 28, we achieve that,

$$\begin{aligned}
\mathbb{I}(\hat{S}^{(k)}) &\geq \hat{\mathbb{I}}(\hat{S}^{(k)}) - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)} \geq (1-\frac{1}{e})\hat{\mathbb{I}}(S^{(k)*}) - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)} \\
&\geq (1-\frac{1}{e})\Big(OPT^{(k)} - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)}\Big) - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}OPT^{(k)} \\
&\geq \Big(1-\frac{1}{e}-(1-\frac{1}{e})\frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}} - \frac{e\epsilon}{2e-1}\frac{1}{(2^i\mu_k^*)^{1/2}}\Big)OPT^{(k)} \\
&\geq \Big(1-\frac{1}{e}-\frac{\epsilon}{(2^i\mu_k^*)^{1/2}}\Big)OPT^{(k)}.
\end{aligned} \tag{30}$$

which happens with probability at least $1 - \frac{(M_k+1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}$ proving the lemma. □

Thus, the above Lemma 4.2 gives a rigorous approximation guarantee on the returned solution in every iteration of LISA. We now prove that at termination point, LISA() returns a solution $\hat{S}^{(k)}$ of size k that meets the $(1 - \frac{1}{e} - \epsilon)$-guarantee.

LEMMA 4.3 (STOPPING CONDITION). *For a set size $k \in [k_{lower}, k_{upper}]$, LISA returns a solution $\hat{S}^{(k)}$ satisfying,*

$$\Pr[\mathbb{I}(\hat{S}^{(k)}) \geq \Big(1 - \frac{1}{e} - \epsilon\Big)OPT^{(k)}] \geq 1 - \frac{(M_k+1)\delta}{\mathcal{M}}. \tag{31}$$

PROOF. Assume that LISA terminates at iteration T, we will prove Eq. 31 in two possible cases when the stopping condition is met: 1) $\deg_{\mathcal{H}}(\hat{S}^{(k_{lower})}) \geq \Lambda$ at iteration T and 2) $T \geq \log_2(\frac{n}{k_{lower}})$ at termination.

Case $\deg_{\mathcal{H}}(\hat{S}^{(k_{lower})}) \geq \Lambda$ at iteration T. $\deg_{\mathcal{H}}(\hat{S}^{(k_{lower})}) \geq \Lambda$ also means that $\deg_{\mathcal{H}}(\hat{S}^{k}) \geq \Lambda, \forall k \in [k_{lower}, k_{upper}]$. From Lemma 4.2, with probability of at least $1 - \frac{(M_k+1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}$, we have $\hat{\mathbb{I}}(\hat{S}^{(k)}) \geq (1 - \frac{1}{e} - \frac{\epsilon}{(2^T\mu_k^*)^{1/2}})OPT^{(k)}$. From Eq. 27 in the proof of Lemma 4.2, this also implies an intermediate result that,

$$\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq \mathbb{I}(\hat{S}^{(k)}) + \frac{e\epsilon}{2e-1}\frac{1}{(2^T\mu_k^*)^{1/2}}OPT^{(k)}. \tag{32}$$

Then, since $\hat{\mathbb{I}}(\hat{S}^{(k)}) = \frac{\deg_{\mathcal{H}}(\hat{S}^{(k)})}{2^T\Upsilon} \geq \frac{\Lambda}{2^T\Upsilon} = \frac{(1+\frac{e}{2(2e-1)}\epsilon)\Upsilon}{2^T\Upsilon} \geq \frac{(1+\frac{e}{2e-1}\epsilon)^{1/2}}{2^T}$ due to the Taylor series $(1+x)^{1/2} = 1 + \frac{x}{2} - o(x)$, the above inequality infers,

$$\frac{(1+\frac{e}{2e-1}\epsilon)^{1/2}}{2^T} \leq \mu_k + \frac{e\epsilon}{2e-1}\frac{1}{(2^T\mu_k^*)^{1/2}}\mu_k^* \tag{33}$$

$$\Rightarrow \quad \frac{(1+\frac{e}{2e-1}\epsilon)^{1/2}}{2^T} \leq \mu_k^* + \frac{e\epsilon}{2e-1}\frac{1}{(2^T\mu_k^*)^{1/2}}\mu_k^* \tag{34}$$

$$\Rightarrow \quad (1+\frac{e}{2e-1}\epsilon)^{1/2} \leq 2^T\mu_k^* + \frac{e\epsilon}{2e-1}\frac{1}{(2^T\mu_k^*)^{1/2}}2^T\mu_k^* \tag{35}$$

$$\Rightarrow \quad 2^T\mu_k^* + \frac{e\epsilon}{2e-1}(2^T\mu_k^*)^{1/2} - (1+\frac{e}{2e-1}\epsilon)^{1/2} \geq 0. \tag{36}$$

Observe that the last inequality is quadratic function of $(2^T\mu_k^*)^{1/2}$, solving the inequality gives,

$$(2^T\mu_k^*)^{1/2} \geq 1 \text{ or } \frac{1}{(2^T\mu_k^*)^{1/2}} \leq 1. \tag{37}$$

Thus, at iteration T, $\frac{1}{(2^T\mu_k^*)^{1/2}} \leq 1$ which implies,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \epsilon)OPT^{(k)}] \leq \Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \frac{\epsilon}{(2^T\mu_k^*)^{1/2}})OPT^{(k)}], \tag{38}$$

and plugging this back into the result in Lemma 4.2 that

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \frac{\epsilon}{(2^T\mu_k^*)^{1/2}})OPT^{(k)}] \leq \frac{(M_k+1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}, \tag{39}$$

we obtain,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \epsilon)OPT^{(k)}] \leq \frac{(M_k + 1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}.$$

Case $T \geq \log_2(\frac{n}{k_{lower}})$ *at iteration* T. In this case, we apparently also have $T \geq \log_2(\frac{n}{k}), \forall k \in [k_{lower}, k_{upper}]$ and thus,

$$(2^T \mu_k^*)^{1/2} \geq (2^{\log_2(\frac{n}{k_{lower}})} \mu_k^*)^{1/2} \geq (\frac{n}{k_{lower}} \mu_k^*)^{1/2} \geq (n)^{1/2} > 1 \tag{40}$$

Hence, we establish the similar result that,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \epsilon)OPT^{(k)}] \leq \frac{(M_k + 1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}. \tag{41}$$

Since the stopping condition is an OR of the two considered cases, union bound gives,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \epsilon)OPT^{(k)}] \leq \frac{2(M_k + 1)\delta}{2\mathcal{M}\log_2(\frac{n}{k_{lower}})}. \tag{42}$$

Lastly, LISA can terminate at any iteration $i = 1, \ldots, \log_2(\frac{n}{k_{lower}})$, thus, taking union bound over all iterations completes the proof,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \epsilon)OPT^{(k)}] \leq \frac{(M_k + 1)\delta \log_2(\frac{n}{k_{lower}})}{\mathcal{M}\log_2(\frac{n}{k_{lower}})} = \frac{(M_k + 1)\delta}{\mathcal{M}}. \tag{43}$$

□

Based on Lemma 4.3, the overall approximation quality of LISA over all seed set sizes is stated in the following theorem.

THEOREM 4.4 (APPROXIMATION GUARANTEE). *Given a probabilistic graph* $\mathcal{G} = (V, E, w)$*, two precision parameters* ϵ, δ *and min / max seed set sizes* k_{lower}, k_{upper}*, LISA algorithm returns a sequence of seed set* $\hat{S}^{(k_{lower})}, \ldots, \hat{S}^{(k_{upper})}$ *satisfying,*

$$\Pr[\mathbb{I}(\hat{S}^{(k)}) \geq (1 - 1/e - \epsilon)OPT^{(k)}, \textbf{\textit{for all}}\ k \in \{k_{lower}, \ldots, k_{upper}\}] \geq 1 - \delta \tag{44}$$

Equivalently, LISA has an InfSpec approximation factor of $(1 - 1/e - \epsilon)$.

PROOF. From Lemma 4.3, for each $k \in [k_{lower}, k_{upper}]$, we have,

$$\Pr[\hat{\mathbb{I}}(\hat{S}^{(k)}) \leq (1 - \frac{1}{e} - \epsilon)OPT^{(k)}] \leq \frac{(M_k + 1)\delta}{\mathcal{M}}. \tag{45}$$

Taking union bound over all seed set sizes $k \in [k_{lower}, k_{upper}]$ and note that $\mathcal{M} = \sum_{k=k_{lower}}^{k_{upper}} (M_k + 1)$, we achieve,

$$\Pr[\mathbb{I}(\hat{S}^{(k)}) \geq (1 - 1/e - \epsilon)OPT^{(k)}, \forall k \in [k_{lower}, k_{upper}]] \geq 1 - \delta. \tag{46}$$

That completes the proof. □

4.2. Complexity Analysis

Time complexity. The Max-Coverage procedure of LISA can be implemented in a linear-time in terms of the total size of all the hyperedges. As we shall show later in the space complexity section, the expected total size of the hyperedges is $O(\Lambda \cdot n)$. Thus, Max-Coverage has an expected time complexity of $O(\Lambda \cdot n)$.

We shall bound the time-complexity of generating hyperedges via the number of edges examined. Keeping track of the maximum degree in the hypergraph is relatively easy and can be done with little additional cost.

LEMMA 4.5. *The expected number of edges examined by LISA is at most* $\Lambda \cdot m$.

PROOF. The proof consists of two parts 1) bound the expected number of hyperedges $m_{\mathcal{H}}$ and 2) estimate the mean number of edges visited per reverse influence sampling.

Number of hyperedes: Let $v^* = \arg\max_{v \in V} \mathbb{I}(v)$, the most influential node. Note that v^* is not necessary the same with $\hat{v}_1$, selected by LISA. Define $Y_j = \min\{|\{v^*\} \cap \mathcal{E}_j|, 1\}|$, a random variable with mean $\mu_Y = \mathbb{I}(v^*)/n$ and $W_j = \sum_{i=1}^{j}(Y_i - \mu_Y)$.

Denote by $T(\Lambda)$ and $T^*(\Lambda)$ the random variables that correspond to the numbers of sampled hyperedges until $deg_{\mathcal{H}}(\hat{S}^{(k_{lower})}) = \Lambda$ and $deg_{\mathcal{H}}(v^*) = \Lambda$, respectively. Since $\deg_{\mathcal{H}}(v^*) \leq \deg_{\mathcal{H}}(\hat{v}_1) \leq \deg_{\mathcal{H}}(\hat{S}^{(k_{lower})})$, $T(\Lambda) = m_{\mathcal{H}} \leq T^*(\Lambda)$ and hence,

$$\mathbb{E}[T(\Lambda)] \leq \mathbb{E}[T^*(\Lambda)].$$

Similar to the case of random variable X_j defined in Eq. 15, the random variables $W_j, \forall j = 1, 2, \ldots$ together form a martingale. Using Martingale Stopping Theorem (Lemma 21 in [Mitzenmacher and Upfal 2005]) with a note that $\mathbb{E}[T^*(\Lambda)] < \infty$ and hence $\mathbb{E}[W_{T^*(\Lambda)}] < \infty$, we have

$$\begin{aligned}
&\mathbb{E}[W_{T^*(\Lambda)}] = \mathbb{E}[W_0] = \mathbb{E}[Y_0 - \mu_Y] = 0 \Leftrightarrow \mathbb{E}\Big[\sum_{j=0}^{T^*(\Lambda)} Y_j - \Lambda\Big] = \mathbb{E}\Big[\sum_{j=0}^{T^*(\Lambda)} Y_j\Big] - \Lambda = 0 \\
\Leftrightarrow &\sum_{t=\Lambda}^{\infty} \mathbb{E}\Big[\sum_{j=0}^{T^*(\Lambda)} Y_j \mid T^*(\Lambda) = t\Big] \Pr[T^*(\Lambda) = t] = \sum_{t=\Lambda}^{\infty} \mathbb{E}\Big[\sum_{j=0}^{t} Y_j\Big] \Pr[T^*(\Lambda) = t] = \Lambda \\
\Leftrightarrow &\sum_{t=\Lambda}^{\infty} \mu_Y t \Pr[T^*(\Lambda) = t] = \mu_Y \sum_{t=\Lambda}^{\infty} t \Pr[T^*(\Lambda) = t] = \Lambda \\
\Leftrightarrow &\ \mu_Y \mathbb{E}[T^*(\Lambda)] = \Lambda. \qquad (47)
\end{aligned}$$

Therefore,

$$\mathbb{E}[m_{\mathcal{H}}] = \mathbb{E}[T(\Lambda)] \leq \mathbb{E}[T^*(\Lambda)] = \frac{\Lambda}{\mu_Y}. \qquad (48)$$

Average number of edges visited per reverse influence sampling: The reverse influence sampling procedure picks a source vertex u uniformly at random. Then for each vertex v, it will examine all in-neighbors of v with a probability $\mathbb{I}(v, u)$, the probability that v can reach to u over all sample graphs of $\mathcal{G}$ (aka the probability that v influences u). Thus the mean number of edges examined by the procedure is

$$\begin{aligned}
&\frac{1}{n}\sum_{u \in V}(\sum_{v \in V} \mathbb{I}(v, u) d^-(v)) = \frac{1}{n}\sum_{v \in V} d^-(v) \sum_{u \in V} \mathbb{I}(v, u) \\
&= \frac{1}{n}\sum_{v \in V} d^-(v)\mathbb{I}(v) \leq \frac{1}{n}\sum_{v \in V} d^-(v)\mathbb{I}(v^*) = \frac{m}{n}\mathbb{I}(v^*) \qquad (49)
\end{aligned}$$

Therefore, the expected number of edges examined by LISA is at most

$$\frac{m}{n}\mathbb{I}(v^*)\frac{\Lambda}{\mu_Y} = m\mu_Y\frac{\Lambda}{\mu_Y} = \Lambda m \qquad (50)$$

This yields the proof. □

THEOREM 4.6. *LISA has expected running time* $O((\log \frac{2}{\delta} + \log \mathcal{M} + \ln \log_2(\frac{n}{k_{lower}}))\frac{1}{\epsilon^2}(m + n))$ *where* $\log \mathcal{M} < k^* \log \frac{n(k_{upper} - k_{lower} + 1)}{k^*}$ *and* $\binom{n}{k^*} \geq \binom{n}{k}, \forall k \in [k_{lower}, k_{upper}]$ *(*$k^* = k_{upper}$ *if* $k_{lower} \leq k_{upper} \leq n/2$*).*

PROOF. Since Max-Coverage has a time complexity $O(\Lambda n)$ and generating hyperedges has an expected runtime $O(\Lambda m)$, it follows that the expected time complexity of LISA is $O(\Lambda(m+n)) = O((\log\frac{2}{\delta} + \log\mathcal{M} + \ln\log_2(\frac{n}{k_{lower}}))\frac{1}{\epsilon^2}(m+n))$.

To evaluate $\log(\mathcal{M})$, we fist have the following,

$$\log(\mathcal{M}) = \log\Big(\sum_{k=k_{lower}}^{k_{upper}}(M_k+1)\Big) = \log\Big(\sum_{k=k_{lower}}^{k_{upper}}(\binom{n}{k}+1)\Big) \tag{51}$$

Since $\binom{n}{k^*} \geq \binom{n}{k}, \forall k \in [k_{lower}, k_{upper}]$,

$$\begin{aligned}
\log\Big[\sum_{k=k_{lower}}^{k_{upper}}(\binom{n}{k}+1)\Big] &\leq \log\Big[\sum_{k=k_{lower}}^{k_{upper}}\binom{n}{k^*}\Big] \\
&\leq \log\Big[(k_{upper}-k_{lower}+1)\binom{n}{k^*}\Big] \\
&\leq \log\Big[(k_{upper}-k_{lower}+1)\Big(\frac{n}{k^*}\Big)^{k^*}\Big] \\
&\leq k^*\log\frac{n(k_{upper}-k_{lower}+1)}{k^*}
\end{aligned}$$

□

Space complexity. Besides an $O(m+n)$ space to hold $\mathcal{G}$, we show that on average only an additional $O((\log\frac{2}{\delta} + \log\mathcal{M} + \ln\log_2(\frac{n}{k_{lower}}))\frac{1}{\epsilon^2}n)$ space is sufficient to hold the hyperedges. Thus, LISA has an expected near linear space complexity.

LEMMA 4.7. *The expected additional space to store all the hyperedges is* $O((\log\frac{2}{\delta} + \log\mathcal{M} + \ln\log_2(\frac{n}{k_{lower}}))\frac{1}{\epsilon^2}n)$.

PROOF. From the proof of Lemma 4.5, the expected number of hyperedges is at most Λ/μ_Y with $\mu_Y = \max_{v\in V}\mathbb{I}(v)/n$. The mean size of a hyperedge can be computed as

$$1/n\sum_{u\in V}\sum_{v\in V}\mathbb{I}(v,u) = 1/n\sum_{v\in V}\mathbb{I}(v) \leq n\mu_Y$$

Therefore, the expected value of the total sizes of all hyperedges is at most

$$\frac{\Lambda}{\mu_Y}\times n\mu_Y = \Lambda n = (\log\frac{2}{\delta} + \log\mathcal{M} + \ln\log_2(\frac{n}{k_{lower}}))\frac{1}{\epsilon^2}n.$$

This completes the proof. □

4.3. Extension to IC model

Our LISA algorithm is easily extended to work on the Independent Cascade (IC) model without effecting the approximation guarantee and complexity order. Differ from the Linear Threshold (LT) model, the edge weight assumption is $0 \leq w(u,v) \leq 1$ (not $\sum_{u\in V} w(u,v) \leq 1, \forall v$ in LT model). IC also operates in rounds, however, the activation criteria is modified to: instead of having a randomly chosen threshold λ_v and v is activated if the total weights form active neighbors exceed λ_v, a node in IC model becomes active through its incoming-edges from the newly activated neighbors. A newly-activated node u will have a single chance of activating its out-going neighbor v and succeed with probability equal the edge weight $w(u,v)$.

As presented in [Kempe et al. 2003], the IC model is also equivalent to a live-edge model and thus, similarly to LT model, corresponds to an RIS sampling procedure [Borgs et al. 2014], termed RIS-IC. By replacing the RIS-LT sampling in LISA with the IC version and following the analysis as for LT model, we obtain the same approximation guarantees (independent with sampling techniques) and complexity results.

THEOREM 4.8. *LISA algorithm for the IC model has an expected running time* $O((\log \frac{2}{\delta} + \log \mathcal{M} + \ln \log_2(\frac{n}{k_{lower}}))\frac{1}{\epsilon^2}(m+n))$ *and returns a sequence of seed sets* $\hat{S}^{(k_{lower})}, \ldots, \hat{S}^{(k_{upper})}$ *that is an* $(1 - 1/e - \epsilon)$ *InfSpec approximate solution.*

5. EXPERIMENTS

In this section, we experimentally evaluate the performance of LISA against the existing state-of-the-art methods, including D-SSA [Nguyen et al. 2016], IMM [Tang et al. 2015], TIM/TIM+ [Tang et al. 2014] on five real-world networks with a wide range of sizes from various disciplines. Since there is no easy way of extending the existing algorithms for InfSpec problem, we have to run these algorithms multiple times for all $k \in \{k_{lower}, \ldots, k_{upper}\}$. The experimental results show that our algorithm can solve the InfSpec problem several orders of magnitudes faster than the runner-up D-SSA.

5.1. Experimental Settings

Table II: Datasets' Statistics

Datasets	*NetHEPT*	*NetPHY*	*Epinions*	*DBLP*	*Twitter*
Nodes	15K	37K	76K	655K	41.7M
Edges	59K	181K	509K	2M	1.5G
Type	undirected	undirected	directed	undirected	directed
Avg. degree	4.1	4.87	13.4	6.1	70.5

Datasets. We perform our experiments in five datasets: NetHEPT, NetPHY, Epinions, DBLP, and Twitter. The basic statistics of these networks are summarized in Table II. *NetHEPT*, *NetPHY* and *DBLP* are collaboration networks taken from the "High Energy Physics - Theory", "Physics" sections of arXiv.org and "Computer Science Bibliography". These undirected networks were frequently used in previous works [Goyal et al. 2010; Goyal et al. 2011b; Chen et al. 2010]. In the networks, nodes and edges represent authors and co-authorship, respectively. The Epinions dataset is the who-trust-whom online social network of a consumer review site Epinions.com. Specially, the largest network is a large portion of *Twitter*, crawled in July 2009 with 41.7 million nodes and 1.5 billion edges [Kwak et al. 2010].

Metrics. For each algorithm, we measure 1) the *spread of influence*, i.e., the expected number of influenced nodes eventually, 2) the running time, and 3) the peak memory consumption. Note that we only need to run LISA once to get the metrics for all different $k = k_{lower}, \ldots, k_{upper}$, in contrast, we have to run the other algorithms for each value of k individually. We terminate algorithms that take more than 24 hours.

Parameters. We set $\epsilon = 0.1$ and $\delta = 1/n$ for LISA, D-SSA, IMM and TIM/TIM+, unless otherwise mentioned. Finally, we validate the spread of influence of the outputted seed sets using EIVA (Section 3) with very high accuracy level: $\epsilon = 0.01$ and $\delta = 1/n$. In our experiments, we target the sets with sizes from 1 to 1000 ($k_{lower} = 1, k_{upper} = 1000$).

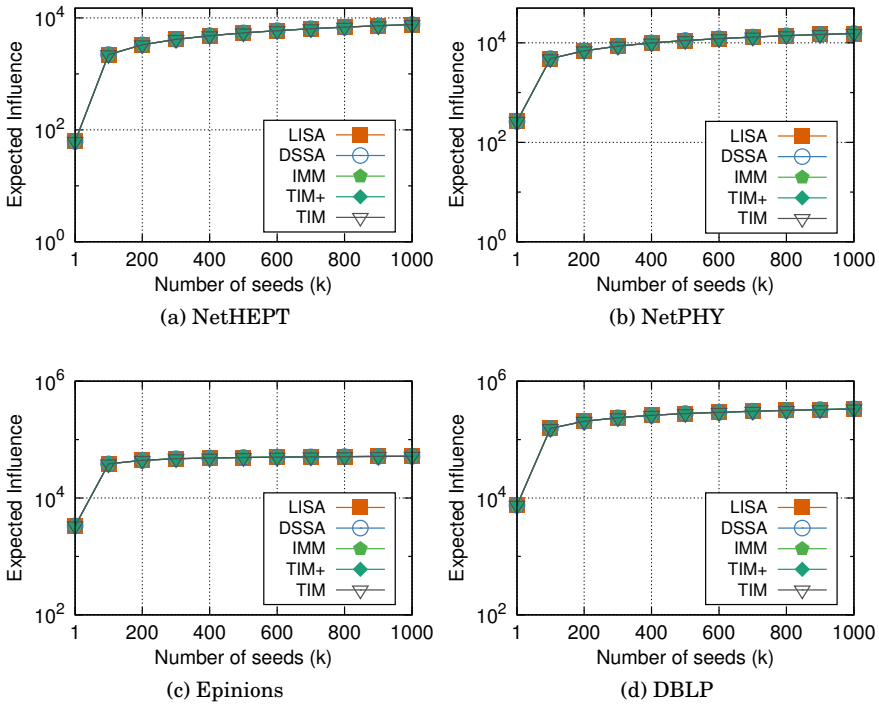


Fig. 1: Spread of Influence under the LT model (the higher the better)

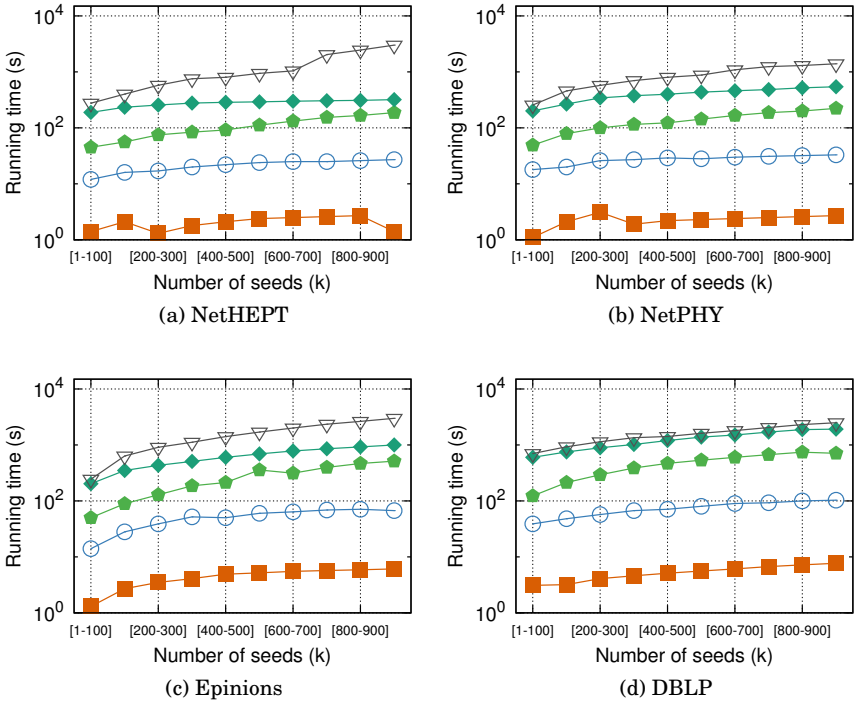
Weight settings. We adopt the common method in [Kempe et al. 2003; Leskovec et al. 2007; Tang et al. 2014; Tang et al. 2015; Nguyen et al. 2016] to calculate the influence weights on edges. More precisely, we assign the weight on an edge (u, v) as $b_{uv} = \frac{A(u,v)}{D(v)}$ where $A(u, v)$ is the number of actions both u and v perform, and $D(v)$ is the in-degree of node v, i.e., $N(v) = \sum_{u \in N^in(v)} A(u, v)$.

Environment. Our implementation is written in C++ and compiled with GCC 4.7. All our experiments are carried out using a Linux machine with a 2.2GHz 8 core Intel Xeon CPU and 100GB memory of RAM.

5.2. Results

We carry two set of experiments: 1) on moderate-size datasets, i.e., NetHEPT, NetPHY, Epinions and DBLP in which we run LISA and the competing algorithms under LT model since the results on IC model are similar and report the expected influence, running time and memory usage; 2) on the billion-scale Twitter network in which we run LISA, D-SSA, IMM, TIM and TIM+ (only these algorithms can handle Twitter dataset) under both the LT and IC models and report the running time and memory usage.

Solution Quality. The quality of the algorithms, measured as the expected number of influenced nodes eventually and termed *expected influence* is shown in Figure 1. We see that all the tested algorithms admit comparable performance in all cases (on all four datasets and with all values of seed set size k). The experimental results also confirm the *viral marketing* behaviors of the influences due to the submodularity property. That is the first few selected nodes carry a huge influence gain and the later ones only



(a) NetHEPT (b) NetPHY

(c) Epinions (d) DBLP

Fig. 2: Running time of the algorithms under the LT model (see Fig. 1 for legends)

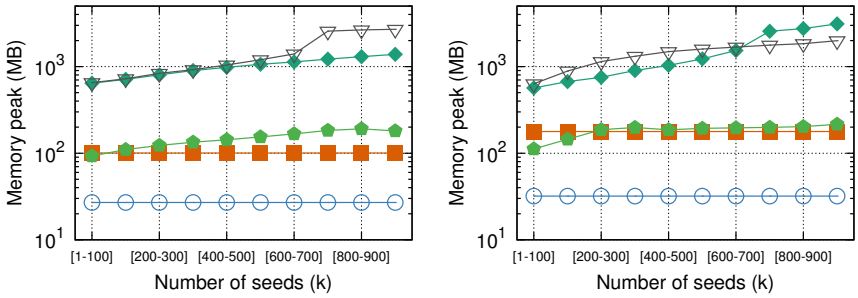
bring in smaller marginal influence. Here, we emphasize that only LISA guarantees all the returned seed sets with sizes up to 1000 to have good quality and thus we only run LISA once with $k_{lower} = 1$ and $k_{upper} = 1000$. The other algorithms can guarantee at each particular size and need to run 1000 times, i.e., one for each value of k.

Running Time. In these experiments, we test the performance of all the algorithms on four moderate-size networks. Since all the methods except LISA have to rerun for each value of $k \in \{k_{lower}, \ldots, k_{upper}\}$, we need to accumulate the times for all runs to get a total running time. Thus, we choose a set of 10 small intervals $(k_{lower}, k_{upper}) \in \{(1, 100), (100, 200), \ldots, (900, 1000)\}$ so that we do not bias and have a fair comparison.

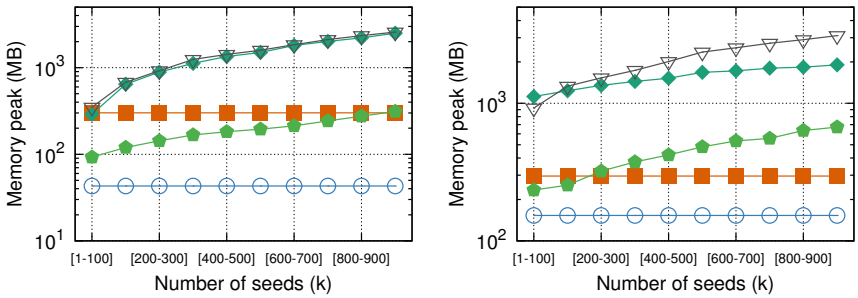
The results are presented in Fig. 2. We see that although the intervals are fairly small, LISA vastly outperform the rest of the algorithms in terms of running time. In particular, LISA is always at least 10 times faster than the second place D-SSA and the speedup is about 100 times compared to IMM. Comparison with TIM/TIM+ shows that LISA is up to three orders of magnitudes faster than these two methods.

Memory Consumption. We show the memory usage of all the algorithms in Fig. 3. We see that LISA consumes much less memory than TIM+ and TIM but more than D-SSA, IMM. However, note that these are moderate-size networks, for larger data as Twitter in the next experiment, LISA requires significantly less memory than IMM, TIM+ and TIM. This experimental observation shows that D-SSA and IMM use less memory implying fewer RIS samples in a particular value of k but require much higher sampling for the whole spectrum of InfSpec.

Experiments on the billion-scale Twitter network. Since Twitter is the largest tested dataset with tens of millions of nodes and billions of edges, we test LISA and

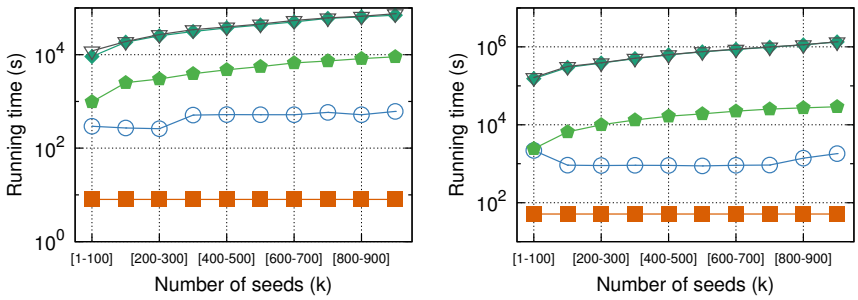


(a) NetHEPT (b) NetPHY



(c) Epinions (d) DBLP

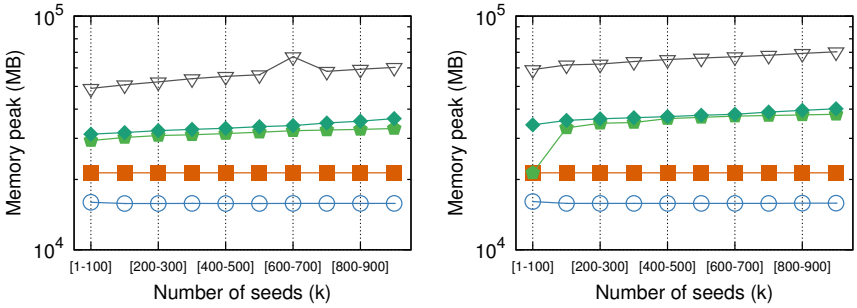
Fig. 3: Memory usage of the algorithms under the LT model (see Fig. 1 for legends)



(a) LT model (b) IC model

Fig. 4: Running time of the best algorithms on the billion-scale Twitter network

other algorithms under both the LT and IC models on this network. Since the solution quality is identical, we only illustrate the running time and memory usage of the algorithms under the LT and IC models. Since D-SSA, IMM, TIM+ and TIM take very long to run Twitter and for smaller k, they requires less time than larger k, we only run them once with $k = k_{lower}$ and consider that to be running time per set size within k_{lower} and k_{upper}. The running time of these algorithms for the spectrum is calculated by multiplying the time per set size with $k_{upper} - k_{lower} + 1$. With LISA, we still run for the whole interval. The results are presented in Figs 4 and 5. These figure again



(a) LT model (b) IC model

Fig. 5: Memory usage of the best algorithms on the billion-scale Twitter network

confirm the superiority of LISA in terms of running time: it is up to several orders of magnitudes faster than the others and requires half of the memory for the others.

6. CONCLUSION

We propose the computation of *Influence Spectrum* (InfSpec) to give better insights for decision making and resource planning in viral marketing campaigns. To compute InfSpec, we design LISA, an efficient approximation algorithm for InfSpec. LISA returns an $(1-1/e-\epsilon)$-approximate influence spectrum with high probability. In practice, LISA also vastly surpasses the state-of-the-art InfMax methods, being in several orders of magnitudes faster than the rest. While the analysis of LISA is based on LT and IC model, all the results also hold the generalized models that combine both LT and IC in [Kempe et al. 2005]. In the future, we will attempt to push the limit further to develop *near linear time approximation algorithms* for InfSpec and InfMax problems.

REFERENCES

C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *Pro. of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*. SIAM, 946–957.

M. Cha, A. Mislove, and K. P. Gummadi. 2009. A measurement-driven analysis of information propagation in the flickr social network. In *WWW '09*. ACM, New York, NY, USA, 721–730.

N. Chen. 2009. On the Approximability of Influence in Social Networks. *SIAM Journal of Discrete Mathematics* 23, 3 (2009), 1400–1415.

Wei Chen, Laks VS Lakshmanan, and Carlos Castillo. 2013. Information and influence propagation in social networks. *Synthesis Lectures on Data Management* 5, 4 (2013), 1–177.

W. Chen, W. Lu, and N. Zhang. 2012. Time-Critical Influence Maximization in Social Networks with Time-Delayed Diffusion Process. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

W. Chen, C. Wang, and Y. Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *ACM KDD '10*. ACM, New York, NY, USA, 1029–1038.

E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. 2014. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM CIKM*. ACM, 629–638.

P. Dagum, R. Karp, M. Luby, and S. Ross. 2000. An Optimal Algorithm for Monte Carlo Estimation. *SIAM J. Comput.* 29, 5 (March 2000), 1484–1496.

T. Dinh, H. Nguyen, P. Ghosh, and M. Mayo. 2015. Social Influence Spectrum with Guarantees: Computing More in Less Time. In *International Conference on Computational Social Networks*. Springer, 84–103.

T.N. Dinh, H. Zhang, D.T. Nguyen, and M.T. Thai. 2014. Cost-Effective Viral Marketing for Time-Critical Campaigns in Large-Scale Social Networks. *Networking, IEEE/ACM Transactions on* (2014).

N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. 2013. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*. 3147–3155.

U. Feige. 1998. A threshold of ln n for approximating set cover. *Journal of ACM* 45, 4 (1998), 634–652.

M. Gomez-Rodriguez, L. Song, N. Du, H. Zha, and B. Schölkopf. 2016. Influence Estimation and Maximization in Continuous-Time Diffusion Networks. *ACM Transactions on Information Systems (TOIS)* 34, 2 (2016), 9.

A. Goyal, F. Bonchi, and LV Lakshmanan. 2010. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 241–250.

A. Goyal, W. Lu, and LV Lakshmanan. 2011a. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th WWW*. ACM, 47–48.

A. Goyal, W. Lu, and LV Lakshmanan. 2011b. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *2011 IEEE 11th ICDM*. IEEE, 211–220.

Z. Huiling, A. A. Md, L. Xiang, T. T. My, and T. N. Hien. 2016. Misinformation in Online Social Networks: Detect Them All with a Limited Budget. *ACM Transactions on Information Systems (TOIS)* 34 (2016).

D. Kempe, J. Kleinberg, and É. Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD'03*. ACM New York, NY, USA, 137–146.

D. Kempe, J. Kleinberg, and E. Tardos. 2005. Influential nodes in a diffusion model for social networks. In *ICALP '05*. 1127–1138.

K. Kutzkov, A. Bifet, F. Bonchi, and A. Gionis. 2013. Strip: stream learning of influence probabilities. In *Proceedings of the 19th ACM SIGKDD*. ACM, 275–283.

H. Kwak, C. Lee, H. Park, and S. Moon. 2010. What is Twitter, a social network or a news media?. In *WWW*. ACM, New York, NY, USA, 591–600.

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. 2007. Cost-effective outbreak detection in networks. In *ACM KDD '07*. ACM, New York, NY, USA, 420–429.

Y. Li, W. Chen, Y. Wang, and Z.L. Zhang. 2013. Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships. In *Pro. of the 6th ACM WSDM*. ACM, 657–666.

L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang. 2010. Mining topic-level influence in heterogeneous networks. In *Proceedings of the 19th ACM CIKM*. ACM, 199–208.

C. Long and R. CW Wong. 2011. Minimizing Seed Set for Viral Marketing. In *Pro. of the 2011 IEEE 11th International Conference on Data Mining*. IEEE Computer Society, Washington, DC, USA, 427–436.

M. Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, J. Stoer (Ed.). Lecture Notes in Control and Information Sciences, Vol. 7. Springer, 234–243.

M. Mitzenmacher and E. Upfal. 2005. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* 14, 1 (1978), 265–294.

D.T. Nguyen, Huiyuan Zhang, S. Das, M.T. Thai, and T.N. Dinh. 2013. Least Cost Influence in Multiplex Social Networks: Model Representation and Analysis. In *2013 IEEE 13th ICDM*. 567–576.

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, New York, NY, USA, 695–710. DOI:http://dx.doi.org/10.1145/2882903.2915207

N. Ohsaka, Y. Akiba, T.and Yoshida, and K. Kawarabayashi. 2014. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *Twenty-Eighth AAAI*.

M. G. Rodriguez and B. Schölkopf. 2012. Influence maximization in continuous time diffusion networks. *arXiv preprint arXiv:1205.1682* (2012).

K. Saito, R. Nakano, and M. Kimura. 2008. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-based intelligent information and engineering systems*. Springer, 67–75.

Y. Shen, T. N. Dinh, H. Zhang, and M. T. Thai. 2012. Interest-matching Information Propagation in Multiple Online Social Networks. In *Pro. of the 21st ACM CIKM*. ACM, New York, NY, USA, 1824–1828.

J. Tang, J. Sun, C. Wang, and Z. Yang. 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD*. ACM, 807–816.

Y. Tang, Y. Shi, and X. Xiao. 2015. Influence maximization in near-linear time: a martingale approach. In *Proceedings of the 2015 ACM SIGMOD*. ACM, 1539–1554.

Y. Tang, X. Xiao, and Y. Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD*. ACM, 75–86.

V.V. Vazirani. 2001. *Approximation Algorithms*. Springer. http://books.google.com/books?id=EILqAmzKgYIC

H. Zhang, T.N. Dinh, and M.T. Thai. 2013. Maximizing the Spread of Positive Influence in Online Social Networks. In *2013 IEEE 33rd ICDCS*. 317–326.

Y. Zhou and L. Liu. 2015. Social Influence Based Clustering and Optimization over Heterogeneous Information Networks. *ACM TKDD* 10, 1 (2015), 2.