A Multi-criteria Approximation Algorithm for Influence Maximization with Probabilistic Guarantees

Maleq Khan* Gopal Pandurangan† Nguyen Dinh Pham‡ Anil Vullikanti§ Qin Zhang¶

Abstract

The well-studied influence maximization problem involves choosing a seed set of a given size, which maximizes the expected influence. However, such solutions might have a significant probability of achieving low influence, which might not be suitable in many applications. In this paper, we consider a different approach: find a seed set that maximizes the influence set size with a given probability. We show that this objective is not submodular, and design a greedy, multi-criteria approximation algorithm for this problem with rigorous approximation guarantees. We also evaluate our algorithm on multiple datasets, and show that they have similar or better quality as the ones optimizing the expected influence, but with additional guarantees on the probability.

1 Introduction

A large number of phenomena, e.g., the spread of influence, fads and ideologies on social networks, can be modeled as a diffusion process on a graph; see, e.g., [7, 13]. From a given seed set S (a subset of vertices), it is assumed that the influence spreads under a well-studied probabilistic diffusion model called the *Independent Cascade (IC)* model (see Sections 1.1 and 2.1); the size of the final influenced set, denoted $I(S)$, is taken as the *influence* of S. An optimization problem that has been extensively studied is the *influence maximization* problem, stated as follows: find a seed set S of size k, so that the expectation, $E[I(S)]$, is maximized—this is referred to as the MaxExpInf problem. The seminal work by Kempe, Kleinberg and Tardos [12] was the first to give a constant-factor approximation to the MaxExpInf problem. Later, Borgs et al. [1] improved the run time by proposing a nearly-optimal algorithm that had the same approximation guarantee. There has been a lot of work on many variants of influence maximization for several diffusion models; see, e.g., [13, 7] for details.

However, there are instances, in which the expected influence set is large, but the variance is also high, which is not desirable. A common way to understand the variance in a random variable is by examining its quantiles. Motivated by this, we focus on the problem of finding a seed set S of size k such that δ - the quantile value (for a given δ, $0 < \delta \leq 1$) of $I(S)$ is maximized — we denote this by $M_\delta(I(S))= \max\{\alpha | \Pr[I(S) \geq \alpha] \geq \delta\}$. The focus of this paper is to find S that maximizes $M_\delta(I(S))$; we refer to this as the MaxProbInf problem.

In this paper, we study the structure of MaxProbInf problem, and develop efficient approximation algorithms for it. Our contributions are the following.

1. We formalize a natural notion of influence maximization with probabilistic guarantees as the quantile value. We also study the structure of solutions to MaxProbInf, and what effect the probabilistic guarantees have.

2. We develop a multi-criteria approximation algorithm, MultiCritMdelta, with approximation guarantees. We also design and analyze an efficient sampling method to estimate $M_\delta(I(S))$ for any given seed set S.

A main technical novelty of our work is the analysis of our multi-criteria approximation algorithm. We use the *Sample Average Approximation* (SAA) technique from stochastic optimization (see, e.g., [19]), which involves constructing samples $G_1, \ldots, G_N$ of the graph, as per the *Independent Cascade* model. Since it is known that $M_\delta(I(S))$ is not submodular [25], we define a different type of submodular function $F_\lambda(S)$, using the *saturation* technique of [14]. We show that finding a minimum cost set S that ensures $F_\lambda(S) \geq \delta\lambda$ is sufficient to give a multi-criteria approximation—this problem is a variant of the standard submodular cover problem. However, the problem does not satisfy an

*Department of Elec. Engg. & Computer Science, Texas A&M University-Kingsville, TX, USA. E-mail: maleq.khan@tamuk.edu.

†Department of Computer Science, University of Houston, Houston, TX 77204, USA. E-mail: gopalpandurangan@gmail.com. Supported, in part, by NSF awards CCF-1527867, CCF-1540512, IIS-1633720, CCF1717075, and BSF award 2016419.

‡Department of Computer Science, University of Houston, Houston, TX 77204, USA. E-mail: aphamdn@gmail.com.

§Dept. of Computer Science, and Biocomplexity Institute, University of Virginia, Charlottesville, VA, USA. E-mail: vsakumar@virginia.edu.

¶Department of Computer Science, Indiana University, Bloomington, IN, USA. E-mail: qzhangcs@indiana.edu. Supported in part by NSF IIS-1633215 and CCF-1844234.

important technical requirement in the submodular cover problem, and we need to modify the analysis of [24] to account for this difference.

3. Finally, we evaluate our methods on several datasets. Not surprisingly, the solutions to $M_\delta(I(S))$ computed using MultiCritMdelta have similar or higher (up to 10% in some cases) quantile values than the MaxExpInf solution. We also observe that the running time of MultiCritMdelta is significantly faster (about 10 times faster) compared to the standard algorithm that implements MaxExpInf (due to Kempe et al [13]) and comparable to the run time of the Borgs et al [1] which is a nearly-optimal algorithm for influence maximization under the expectation measure. In fact, we also observe that our solutions have similar or better *expected influence value*, than the MaxExpInf solution (though this objective was not being optimized); additionally, we get bounds on the probability. In particular, we find that for many instances with low activation probabilities, the seed set output by MultiCritMdelta yields even better (by up to 10%) expected influence value than the MaxExpInf solution, computed using the approximation algorithm of [12] or [1] which both give a constant-factor approximation to the optimal expected influence.

1.1 Related Work

The works by [12], [13], were the first to formulate the problem of influence spreading as a discrete optimization problem. They consider two main diffusion models: *Independent Cascade* (which we adopt in this paper) and *Linear Threshold*. Works in these models include two main optimization problems: maximizing the expected influence with constraints on the seed set (usually size), and minimizing the seed set (according to some measurement) to achieve a target expected influence. These problems are at least NP-hard, [13]. In [4], it was shown that computing the expected influence is #P-hard.

The approach proposed by [12], using submodular set function and greedy heuristic, gives a $(1 - 1/e - \epsilon)$-approximation to the maximizing the expected influence problem. The error ϵ is due to Monte Carlo estimation of the expected influence, which is the main issue in practice, where large real world graphs discourage excessive sampling. Borgs et al. [1] propose an algorithm with nearly optimal theoretical runtime of $O((m+n)k\epsilon^{-2}\log n)$ while retaining the same approximation guarantee. The technique is to sample *reversed* influence, which is adopted and improved upon in other works, e.g., [21], [20], [18]. While reversed influence sampling has a theoretical guarantee for the expectation problem, it is not extendable to our probabilistic $M_\delta(I(S))$. Another approach, proposed by [16], is to estimate expected influence as a Riemann sum, using $O(n\epsilon^2\text{polylog}(n))$ samples which can be implemented in parallel by using MapReduce.

Other models are also studied, for example, fixed threshold models [2], [10], time-restricted diffusion model [11], [3], [5], continuous-time diffusion model [6], and diffusion in dynamic network [23].

The work closest to ours is presented by Zhang et al. [25]. They introduce the *Seed minimization with probabilistic coverage guarantee* (SM-PCG) problem: find the smallest seed set S that ensures that $\Pr[I(S) \geq \eta] \geq P$, where η and P are parameters. They also give an additive approximation to the minimum seed set size needed for a given η, P, and show that the solutions achieve similar expected influence, but with the additional guarantee on the probability. They show that the size of the output seed set, compared to the optimal one, incurs both a multiplicative error of $(\ln n + O(1))$ and an additive error of $O(\sqrt{n})$, *under the assumption* that the standard deviation of the influence is $O(\sqrt{n})$. In contrast, our goal is different — we want to maximize influence with probabilistic guarantee, with a constraint on the seed set size. Our multi-criteria approximation does not incur additive error, and does not rely on assumption about the distribution. There are two limitations of their work. First, the additive approximation is $O(\sqrt{n})$, which can be quite large. In contrast, influence maximization has typically been studied for bounded seed set sizes, which, ideally, should be small. Second, the achievable influence η is not known a priori, and so the algorithm would have to be run for multiple η values to understand a tradeoff.

2 Preliminaries

2.1 Model and Problem Definition

Consider a graph G with n nodes and m directed edges. This graph models a network of influence, where an edge (u, v) has a weight (probabiltiy) $0 \leq p(u, v) \leq 1$, indicating how likely u influences v. The problem of interest is to analyze how influence is propagated in the network. We use the well-studied *Independent Cascades (IC) model* with discrete time to model the spread of influence which we explain below [12].

At any given time t, the nodes have one of three states: *active, newly active, inactive*. At time t, let A_t be the set of active nodes, S_t be the set of newly active nodes, and the rest be inactive. Each node $u \in S_t$ can activate each of its inactive neighbors v with a probability $p(u, v)$. Let U_t be the set of nodes activated in this step. At time $t+1$, $A_{t+1} = A_t \cup U_t$, and $S_{t+1} = U_t$. Starting from an initial configuration of a *seed set* $S = S_0$, we apply the process until time τ where $U_\tau = \varnothing$, indicating no new nodes can be activated.

We denote $I(S) = |A_\tau|$ as the *influence* of the set S. More generally, we have a weight w_v for each node v, and $w(I(S)) = \sum_{v \in A_\tau} w_v$ is the total weight of the influence set $I(S)$. For simplicity, we will focus on the unweighted version of the problem; all our results hold for the weighted version as well, with natural changes in bounds. We note that $I(S)$ is a random variable that depends on S (and the underlying diffusion process).

As mentioned in Section 1, the MaxExpInf problem, maximizing $E[I(S)]$, is a coarse optimization. In particular, it does not give probabilistic guarantees on the influence of the chosen seed set. To get a better idea of $I(S)$, we may need to estimate the variance, or even to acquire the distribution of $I(\cdot)$. However, this task is usually quite difficult and costly.

In this paper, we propose another measure which could be more useful: given a threshold probability δ, find a seed set S such that the δ-quantile value of $I(S)$ is maximized. This measure gives direct probabilistic guarantees on the random variable $I(S)$.[1] More formally, for some set S, and a threshold δ, define the following measure:

$$M_\delta(I(S)) = max\{a | \Pr[I(S) \geq a] \geq \delta\} \tag{2.1}$$

The new optimization problem, referred to as MaxProbInf is defined in the following manner: given an instance $(G = (V, E), B, w, \delta)$, find a (seed) set S such that we:

$$\begin{aligned} &\text{maximize} && M_\delta(I(S)) \\ &\text{subject to} && \sum_{i \in S} w_i \leq B. \end{aligned} \tag{2.2}$$

The goal of this paper is to study the above optimization problem and design algorithms for it. In our analysis, the omitted proofs are given in the appendix.

2.2 Comparison with MaxExpInf As mentioned, the standard influence maximization problem, MaxExpInf, is defined as following [13]: given an instance $(G = (V, E), k)$, find a set $S \subseteq V$ of size at most k such that $E[I(S)]$ is maximized.

A natural question is whether one could find a solution to the problem of maximizing $M_\delta(I(S))$, i.e., MaxProbInf by solving MaxExpInf. We show below that, in general, the solutions of these two problems can be quite different.

Lemma 2.1. *There exist instances (G, k, δ), for which $\frac{E[I(S^*)]}{M_\delta(I(S^*))}$ is arbitrarily large, where S^* is an optimum solution to the* MaxExpInf *problem.*

[1] Note that one can obtain a one-sided probability bound from expectation using Markov's inequality; but this usually quite weak.

Proof. In appendix. ∎

2.3 Hardness We observe that MaxProbInf is NP-hard to approximate within a factor of $(1 - 1/e)$, which is similar to the hardness of the MaxExpInf problem.

Lemma 2.2. *It is NP-hard to obtain an approximate solution to the* MaxProbInf *problem, within a factor of $(1 - 1/e)$.*

Proof. In appendix. ∎

3 A Multi-criteria Approximation Algorithm for Computing MaxProbInf

Motivated by the hardness from Lemma 2.2, and the non-submodularity of $M_\delta(I(S))$ [25], we consider a multi-criteria approximation algorithm, which relaxes the seed set size k, as well as the probability parameter δ. Specifically, we consider the following variant of the MaxProbInf problem: find S such that $M_\delta(I(S)) \geq \gamma M_{\delta'}(I(S^*))$ and $|S| \leq \beta k$, where S^* is an optimal solution, and $\gamma < 1, \beta > 1$ are the relaxation parameters, and $\delta' > \delta$.

3.1 Using submodularity and the sample average approximation technique MaxProbInf is a stochastic optimization problem. We reduce this to a deterministic problem using the *sample average approximation* technique: assume we have N samples $G_1, \ldots, G_N$ of the graph $G = (V, E)$; $G_i = (V, E_i)$ is the ith sample, and is obtained by picking each edge $e \in E$ independently with probability $p(e)$. Let $F_i(S)$ denote the total influence due to S in graph G_i – this equals the sum of the sizes of the components containing nodes in S in sample G_i. For any $S \subseteq V$, let $h_\delta(S)$ be the δ-quantile value estimated from these samples in the following manner: suppose the samples are ordered so that $F_1(S) \geq \ldots \geq F_N(S)$. Then $h_\delta(S) = F_{\lfloor \delta N \rfloor}(S)$. Let S^* be the optimum solution that maximizes $M_\delta(I(S))$.

Lemma 3.1. *Given $\delta \in (0, 1)$, then for any $\epsilon \in (0, 1)$, there exists $N = \Omega(n \log n)$ such that $h_\delta(S) \in [M_{\delta(1+\epsilon)}(I(S)), M_{\delta(1-\epsilon)}(I(S))]$, for all $S \subseteq V$, with high probability.*

Proof. Let us fix some set S. Let L be the number of F_i which is greater than the $(1 - \epsilon)\delta$ quantile, and let R be the numberer of F_i which is greater than the $(1 + \epsilon)\delta$ quantile. We have: $\mu_L = E[L] = (1 - \epsilon)\delta N$ and $\mu_R = E[R] = (1 + \epsilon)\delta N$. $h_\delta(S)$ will fall outside the desired range if either $L > (1 + \epsilon)\delta N$ or $R < (1 - \epsilon)\delta N$. Let $\mathcal{E}_S$ indicates such failure event. Probability of failure is:

$$Pr[\mathcal{E}_S] \leq Pr[L > \mu_L(1 + \frac{2\epsilon}{1 - \epsilon})] + Pr[R < \mu_R(1 - \frac{2\epsilon}{1 + \epsilon})]$$

Applying Chernoff bound:

$$Pr[\mathcal{E}_S] \leq \left(\frac{e^{2\epsilon/(1-\epsilon)}}{(\frac{1+\epsilon}{1-\epsilon})^{\frac{1+\epsilon}{1-\epsilon}}}\right)^{(1-\epsilon)\delta N} + \left(\frac{e^{-2\epsilon/(1+\epsilon)}}{(\frac{1-\epsilon}{1+\epsilon})^{\frac{1-\epsilon}{1+\epsilon}}}\right)^{(1+\epsilon)\delta N}$$

Since the bases of the exponents are in the range $(0,1)$ for $0 < \epsilon < 1$, we have:

$$Pr[\mathcal{E}_S] \leq e^{-cN} \quad \text{for some } c > 0 \tag{3.3}$$

With 2^n possible set S, let $\mathcal{E}_{2^n}$ indicates the event that at least one set has the wrong $h_\delta(.)$ estimation. By union bound:

$$Pr[\mathcal{E}_{2^n}] \leq e^{-cN} 2^n = e^{-cN + n \ln 2} \tag{3.4}$$

Thus, when $N = \Omega(n \log n)$, with probability of $1 - O(n^{-1})$, all $h_\delta(S)$ are correctly estimated. ∎

It turns out approximating the function $h_\delta(S)$ directly is hard; in particular, it is not submodular, and thus a greedy strategy might not work. Instead, we will consider a slight variant, using ideas from the result of Krause et al. [14] on minimum cost submodular cover problem. Let λ be a "guess" of $M_\delta(I(S^*))$, where S^* is an optimal solution. Define the truncated function $F_{i,\lambda}(S) = \min\{F_i(S), \lambda\}$, and

$$F_\lambda(S) = \frac{1}{N} \sum_i F_{i,\lambda}(S).$$

A critical observation is that the truncated functions are submodular, formally:

LEMMA 3.2. *(see [9]) The functions $F_{i,\lambda}(S)$ and $F_\lambda(S)$ are monotone and submodular.*

We show that maximizing $F_\lambda(S)$ is good enough for our purpose, due to the following property.

LEMMA 3.3. *If $h_\delta(S) \geq \lambda$, then $F_\lambda(S) \geq \delta\lambda$. On the other hand, if $F_\lambda(S) \geq \delta\lambda$, then $h_{\delta/2}(S) \geq \delta\lambda/2$.*

Proof. The "if" part: let $A = \{i : F_{i,\lambda}(S) \geq \lambda\}$. Since $h_\delta(S) \geq \lambda$, it follows that $|A| \geq \delta N$. This implies

$$F_\lambda(S) = \frac{1}{N} \sum_i F_{i,\lambda}(S) \geq \frac{1}{N} \sum_{i \in A} F_{i,\lambda}(S) \geq \frac{1}{N}(\lambda\delta N) = \lambda\delta$$

The "only if" part: let $B = \{i : F_{i,\lambda}(S) \geq \delta\lambda/2\}$. We have:

$$\lambda|B| + \sum_{i \notin B} \delta\lambda/2 \geq \sum_{i \in B} F_{i,\lambda}(S) + \sum_{i \notin B} F_{i,\lambda}(S) = F_\lambda(S)N \geq \delta\lambda N.$$

This implies

$$\lambda|B| + (N - |B|)\delta\lambda/2 \geq \delta\lambda N,$$

so that

$$|B|\lambda(1 - \delta/2) \geq N\delta\lambda/2$$

Therefore,

$$|B| \geq \frac{N\delta/2}{1 - \delta/2} \geq N\delta/2$$

This implies $h_{\delta/2}(S) \geq \delta\lambda/2$. □ ∎

Algorithm 1 Algorithm SIMPLEGREEDY

```
function SimpleGreedy(G(V,E), w, δ', n, N)
    for each guess λ do
        S_λ ← ∅
        C ← δ'λ
        while F_λ(S_λ) < C do
            Pick node j that minimizes w_j / (F_λ(S_λ ∪ {j}) − F_λ(S_λ))
            S_λ ← S_λ ∪ {j}
    return S_λ = S_{λ,δ'} that maximizes F_λ(S_λ)
```

Lemma 3.3 motivates the following strategy in order to maximize $h_\delta(\cdot)$ (which is sufficient for solving MAXPROBINF, due to Lemma 3.1): find S such that $F_\lambda(S) \geq \delta\lambda$ and $\text{cost}(S) = \sum_{j \in S} w_j$ is minimized, where w_j is the weight of node j. This is a variant of the minimum cost submodular cover problem [8], which involves finding a minimum cost set S such that $F(S) = F(V)$ for a submodular function $F(\cdot)$, where V is the universe.

Algorithm 1 describes SIMPLEGREEDY, which guesses λ, and greedily computes a set S_λ. This algorithm is the same as that for the standard minimum cost submodular cover problem, except that the stopping condition is $F_\lambda(S_\lambda) \geq \delta\lambda$, instead of $F_\lambda(S_\lambda) = F_\lambda(V)$. We show, however, that the same guarantee can be obtained by appropriately modifying the analysis of the minimum cost submodular cover problem.

For a constant $\delta' \in (0,1)$, let $S^*_{\delta'} = \text{argmax}_{S, cost(S) \leq k} h_{\delta'}(S)$, and $\lambda^*_{\delta'} = h_{\delta'}(S^*_{\delta'})$.

LEMMA 3.4. *Let $S^*_{\delta'}$ and $\lambda^*_{\delta'}$ be as defined above. Let set S_λ be the solution returned by algorithm* SIMPLEGREEDY *for some λ, when the probability parameter is set to δ'. Then, we have: (1) $\lambda \geq \lambda^*_{\delta'}$, (2) $F_\lambda(S_\lambda) \geq \lambda^*_{\delta'}\delta'$, and (3) $cost(S) = O(\ln(N\lambda))cost(S^*)$.*

Our proof of Lemma 3.4 is a variation of the proof by Wolsey [24]. For notational simplicity, we drop the subscript λ in $F_\lambda(S)$ below. Let $\rho_j(S) = F(S \cup \{j\}) - F(S)$. First observe that the following IP is feasible: (IP) $\min \sum_j w_j x_j$ such that for all $S \subseteq V$, with $x_j \in \{0,1\}$ we have $\sum_{j \notin S} \rho_j(S) x_j \geq C - F(S)$.

LEMMA 3.5. *The program (IP) is valid, i.e., the optimum solution S_{IP} satisfies $F(S_{IP}) \geq C$ and S_{IP} has the minimum cost.*

The dual program of the linear relaxation of (IP) is the following

$$\text{(D) max} \quad \sum_S (C - F(S))y_S \quad \text{such that}$$

$$\begin{aligned} \sum_{S:j\notin S} \rho_j(S)y_S &\leq w_j \text{ for all } j \\ y_S &\geq 0 \end{aligned}$$

Observe that the algorithm actually picks element j which minimizes $\frac{w_j}{\rho_j(S)}$. We construct an approximate dual solution. Suppose the greedy algorithm picks elements $\{j_1, \ldots, j_\ell\}$. Let $S_i = \{j_1, \ldots, j_i\}$. Define

$$\theta_i = \begin{cases} \frac{w_{j_i}}{F(S_i)-F(S_{i-1})} = \frac{w_{j_i}}{\rho_{j_i}(S_{i-1})}, \text{ for } i < \ell, \\ \frac{w_{j_\ell}}{C-F(S_{i-1})}, \text{ for } i = \ell. \end{cases}$$

Define

$$y_S = \begin{cases} \theta_1, \text{ if } S = S_0 = \emptyset, \\ \theta_{i+1} - \theta_i, \text{ if } S = S_i \text{ for } 0 < i < \ell, \\ 0, \text{ otherwise.} \end{cases}$$

Observe that the dual solution y_{S_i} covers the cost of the solution S_ℓ:

$$\begin{aligned} \sum_S (C - F(S))y_S &= \sum_{i=0}^{\ell-1} (C - F(S_i))y_{S_i} \\ &= \sum_{i=0}^{\ell-1} (C - F(S_i))(\theta_{i+1} - \theta_i) \\ &= \sum_{i=1}^{\ell-1} \theta_i(F(S_i) - F(S_{i-1})) + \theta_\ell(C - F(S_{\ell-1})) \\ &= \sum_{i=1}^{\ell-1} w_{j_i} + w_{j_\ell} = \text{cost}(S_\ell). \end{aligned}$$

Next, we observe that the dual constraints are approximately feasible. First, consider any $j \in V - S_\ell$. We have

$$\begin{aligned} \sum_{S:j\notin S} \rho_j(S)y_S &= \sum_{i=0}^{\ell} \rho_j(S_i)y_{S_i} \\ &= \rho_j(S_0)\theta_1 + \sum_{i=1}^{\ell-1} \rho_j(S_i)(\theta_{i+1} - \theta_i) \\ &= \sum_{i=1}^{\ell-1} \theta_i(\rho_j(S_{i-1})) + \theta_\ell\rho_j(S_{\ell-1}) \\ &\leq \sum_{i=1}^{\ell-1} w_j \frac{\rho_j(S_{i-1}) - \rho_j(S_i)}{\rho_j(S_{i-1})} + w_j, \end{aligned}$$

because $\theta_i \leq \frac{w_j}{\rho_j(S_{i-1})}$ by construction, for each $i \leq \ell$.

For $x \in (0, \rho_j(\emptyset))$, define $h(x) = \frac{1}{\rho_j(S_{i-1})}$, if $\rho_j(S_i) < x \leq \rho_j(S_{i-1})$. Let

$$\delta = \min_{S:\rho_j(S)>0} \rho_j(S) = \min_{S:\rho_j(S)>0} F(S+j) - F(S) \geq \frac{1}{N}.$$

Therefore,

$$\begin{aligned} &\sum_{i=1}^{\ell-1} \frac{\rho_j(S_{i-1}) - \rho_j(S_i)}{\rho_j(S_{i-1})} = \int_0^{\rho_j(\emptyset)} h(x)dx \\ &\leq \int_0^{\delta} \frac{1}{\delta}dx + \int_\delta^{\rho_j(\emptyset)} \frac{1}{x}dx = 1 + \ln\frac{\rho_j(\emptyset)}{\delta} \leq 1 + \ln(N\lambda), \end{aligned}$$

since $\rho_j(\emptyset) \leq F(j) \leq \lambda$. This implies

$$\sum_{S:j\notin S} \rho_j(S)y_S \leq (2 + \ln(N\lambda))w_j.$$

Next, suppose $j \in S_r$ for some r. Then, $\rho_j(S_r) = F(S_{r+1}) - F(S_r) = 0$. Let $r' < r$ be the largest index such that $\rho_j(S_{r'}) > 0$. In that case, the dual constraint for j can be written as

$$\begin{aligned} \sum_{S:j\notin S} \rho_j(S)y_S &= \sum_{i=0}^{r'} \rho_j(S_i)y_{S_i} \\ &= \sum_{i=1}^{r'+1} \theta_i(\rho_j(S_{i-1}) - \rho_j(S_i)) \\ &\leq \sum_{i=1}^{r'+1} w_j \frac{\rho_j(S_{i-1})\rho_j(S_i)}{\rho_j(S_{i-1})} \leq 1 + \ln(N\lambda), \end{aligned}$$

as before. Therefore, $\frac{1}{\alpha}y$ is a feasible solution, for $\alpha = 2 + \ln N\lambda$, which completes the proof for Lemma 3.4. □

Finally, we put everything together to obtain an approximation to the MAXPROBINF problem.

THEOREM 3.1. *Let $\delta \in (0,1)$ be a constant, and let k be a parameter. Let S_δ^{opt} denote an optimum solution to the* MAXPROBINF *problem for parameter δ, i.e., $M_\delta(I(S^{opt})) \geq M_\delta(I(S'))$, for all $|S'| \leq k$. For any $\epsilon \in (0,1)$, there exists $N = \Omega(n \log n)$ such that if set S is the solution computed by algorithm* SIMPLEGREEDY *with parameter $\delta' = 2\delta/(1-\epsilon)$, we have $M_\delta(I(S)) \geq \frac{\delta}{1-\epsilon} M_{2\delta(1+\epsilon)/(1-\epsilon)}(I(S^{opt}_{2\delta/(1-\epsilon)}))$, and $|S| = O(k \log n)$ The running time of* SIMPLEGREEDY *is $O(n^2 mk \log^2 n)$.*

Proof. Using Lemma 3.4, for the unweighted case, $cost(S) = |S|$. Let $S^*_{\delta'} = \text{argmax}_{S,|S|\leq k} h_{\delta'}(S)$, and $\lambda^*_{\delta'} = h_{\delta'}(S^*_{\delta'})$ defined similarly. We have $\lambda \geq \lambda^*_{\delta'}$,

Algorithm 2 k-Influence set by MULTICRITMDELTA

function MULTICRITMDELTA($G(V,E), k, \delta, n, N$)
  $l \leftarrow 1$
  $h \leftarrow n$
  for *step* $\leftarrow [1 \dots \log n]$ **do**
    $S \leftarrow \emptyset$
    $\lambda \leftarrow (l+h)/2$
    $C \leftarrow \delta\lambda$, and $F(S) \leftarrow F_\lambda(S)$
    while $F(S) < C$ **do**
      Pick node j that minimizes $\frac{w_j}{F(S\cup\{j\})-F(S)}$
      $S \leftarrow S \cup \{j\}$
    feasible if $|S| \leq k\ln(N\lambda) : l \leftarrow \lambda$
    infeasible if $|S| > k\ln(N\lambda)$: $h \leftarrow \lambda$
  return S

and $F_\lambda(S') \geq \lambda\delta'$. By Lemma 3.3, this implies that $h_{\delta'/2}(S) \geq \lambda\delta'/2 \geq \lambda^*_{\delta'}\delta'/2 = \frac{\delta'}{2}h_{\delta'}(S^*_{\delta'})$.

By Lemma 3.1, we have $M_{\delta'(1-\epsilon)/2}(I(S)) \geq h_{\delta'/2}(S)$. Next, $h_{\delta'}(S^*_{\delta'}) \geq h_{\delta'}(S^{opt}_{\delta'})$, by definition of $S^*_{\delta'}$. Again, by Lemma 3.1, we have $h_{\delta'}(S^{opt}_{\delta'}) \geq M_{\delta'(1+\epsilon)}(I(S^{opt}_{\delta'}))$. Combining all these, we get

$$M_{\delta'(1-\epsilon)/2}(I(S)) \geq \frac{\delta'}{2}M_{\delta'(1+\epsilon)}(I(S^{opt}_{\delta'}))$$

Setting $\delta' = 2\delta/(1-\epsilon)$, we get the approximation guarantee $M_\delta(I(S)) \geq \frac{\delta}{1-\epsilon}M_{2\delta(1+\epsilon)/(1-\epsilon)}(I(S^{opt}_{2\delta/(1-\epsilon)}))$.

The algorithm may need to check all n possible values of λ, each check constructs a set of size $O(k\log n)$. The number of samples $N = \Omega(n\log n)$ is sufficient for high probability of success, by adjusting the union bound from equation 3.4:

$$Pr[Failure] \leq e^{-cN}n2^n k\log n = ke^{-cN+\ln n+n\ln 2+\ln\log n} = 1 - O(n^{-1})$$

The complexity for one sample is $O(m)$, and total time complexity is: $O(Nnmk\log n) = O(n^2 mk\log^2 n)$.

□

3.2 Fast implementation of algorithm Multi-CritMdelta. The pseudocode is shown in Algorithm 2. To improve the running time, we use binary search as follows. With a given λ, an iteration decides if it is feasible to construct S such that $F_\lambda(S) = C = \delta\lambda$, with a log factor constraint on the size of S. If it is the case, λ is a feasible value, and S is a feasible solution for $maxF_\lambda(\cdot)$ problem. Starting with $\lambda = n/2$, the binary search increases or decreases λ as per the feasibility. The output is a feasible value and the respective set. Lemma 3.4 shows the approximation guarantee of this algorithm.

4 Computing $M_\delta(I(S))$ for a fixed set S

In this section, we show how to compute an approximation of $M_\delta(I(S))$ for a *fixed* set S, efficiently. It is known that computing $M_\delta(I(S))$ exactly even for a fixed set S is #P-Hard [25]. However, we show that we can estimate $M_\delta(I(S))$ by using Monte-Carlo sampling.

4.1 Monte-Carlo approximation of $M_\delta(.)$ Firstly, it will be useful to consider the following general problem of estimating $M_\delta(.)$ for a random variable that takes values between 0 and 1 (both included).

$$\begin{array}{ll} 0 \leq X \leq 1 & \text{a random variable,} \\ 0 < \delta \leq 1 & \text{a probability threshold,} \\ \text{find} & M_\delta(X) = sup\{a|\Pr[X \geq a] \geq \delta\}. \end{array} \quad (4.5)$$

We define $M_\delta(X)$ in a way that is valid for both discrete and continuous distributions. Indeed, let F_X be the cumulative distribution of X, if F_X is continuous, we can define: $M_\delta(X) = sup\{a|\Pr[X \geq a] = \delta\}$, and the solution is given by: $M_\delta(X) = F_X^{-1}(1-\delta)$.

In general, we cannot afford finding the distribution function, thus, we apply a Monte Carlo sampling to give an (ν,ϵ,ζ)-approximation of $M_\delta(X)$.

DEFINITION 4.1. *$\widetilde{M}_\delta(X,\nu,\epsilon,\zeta)$ is an (ν,ϵ,ζ)-approximation of $M_\delta(X)$ if the following inequality holds, with (confidence) probability $1-\zeta$,*

$$M_{\delta+\eta}(X) - \nu \leq \widetilde{M}_\delta(X,\nu,\eta,\zeta) \leq M_{\delta-\eta}(X) + \nu \quad (4.6)$$

Notice that both the approximation errors η and ν are additive, i.e. absolute values. In the general setting, an additive error is less desirable to a multiplicative error, since we do not know the order of the estimated parameter in advance. We need the absolute errors for the design and analysis of our efficient sampling method, and we observe that this is not an issue for the specific influence problem. First, δ is a given constant, and we can convert between multiplicative and additive error, $\epsilon = \frac{\eta}{\delta}$, or $\eta = \delta\epsilon$. Second, for the influence problem, X is a discrete random variable in $[0,n]$, thus we can set $\nu = \frac{1}{n}$ to have a tighter bound.

We now describe the estimation by sampling. The idea is to perform binary search to find the best value of a, the approximation. Initially, we guess $a = 1/2$, then verify if $\Pr(X \geq a) \geq \delta$. If the test is confirmed, we make the next guess as $a = 3/4$, otherwise, we guess $a = 1/4$. Continue until the step of the guess is smaller than ν.

Consider one iteration, with some fixed a, we need an estimation for the unknown probability $p = \Pr(X \geq a)$. We proceed similarly to the proof of Lemma 3.1, with a different form of Chernoff bound, which is more

convenient to bound the number of samples. Take N samples $X_1, X_2, \cdots, X_N$, where N will be specified later. Define N indicator random variables Z_i, where $Z_i = 1$ if the sample $X_i \geq a$, 0 otherwise. Let $Z = \sum_{i=1}^{N} Z_i$. We have $\mu_Z = \mathrm{E}[Z] = Np$. Using Chernoff bound[17], we bound the empirical probability $\bar{p} = \frac{Z}{N}$, within some multiplicative error κ:

$$\Pr(|Z - Np| \geq \kappa Np) \leq 2exp\left(-\frac{\kappa^2}{2+\kappa}Np\right)$$
$$\iff \Pr\left(|\bar{p} - p| \geq \kappa p\right) \leq 2exp\left(-\frac{\kappa^2}{2+\kappa}Np\right).$$

With the desired error $\eta = \kappa p \iff \kappa = \frac{\eta}{p}$, the tail bound becomes:

(4.7)
$$\Pr\left(|\bar{p} - p| \geq \eta\right) \leq 2exp\left(-\frac{\eta^2}{2p+\eta}N\right) \leq 2exp\left(-\frac{\eta^2 N}{3}\right).$$

Given $|\bar{p} - p| \geq \eta$, clearly we have: $\bar{p} - \eta \geq \delta \implies p \geq \delta$, and $\bar{p} + \eta < \delta \implies p < \delta$. This method leaves an undecided region when $\bar{p} - \eta < \delta \leq \bar{p} + \eta$. As we allow η error around δ, we can simplify the decision rule:

(4.8)
$$\begin{aligned} &\text{If } \bar{p} \geq \delta \implies \text{increase } a; \\ &\text{If } \bar{p} < \delta \implies \text{decrease } a. \end{aligned}$$

Algorithm 3 shows the pseudo code for the approximation. The following lemma states the guarantee bound as per equation 4.6.

LEMMA 4.1. *Assuming algorithm 3 is successful, it returns an (ν, η, ζ)-approximation of $M_\delta(X)$.*

Proof. The decision rule in equation 4.8 ensures this invariance: if the guess increases then at least $p \geq \delta - \eta$, if the guess decreases then at least $p \leq \delta + \eta$. This invariance holds for every iteration. At the final iteration, the bounds are offset by $\pm\nu$ by the precision of the binary search, taking the conservative case, we have the approximation 4.6. ∎

The next lemma states the number of samples required for succeeding with probability at least $1 - \zeta$.

LEMMA 4.2. *Algorithm 3 is successful with probability of at least $(1 - \zeta)$ using $N \geq \frac{3}{\eta^2}\log\nu^{-1}(\ln(\log\nu^{-1}) + \ln(2\zeta^{-1}))$ samples.*

Proof. Let N_{iter} be the number of samples in one iteration. From the bound in equation 4.7, using union bound for $\log\frac{1}{\nu}$ iterations, we require:

$$2exp\left(-\frac{\eta^2 N_{iter}}{3}\right)\log\frac{1}{\nu} \leq \zeta$$
$$\iff \frac{\eta^2 N_{iter}}{3} \geq \ln\log\frac{1}{\nu} + \ln\frac{2}{\zeta}$$

Thus the total number of samples is:

$$N = N_{iter}\log\frac{1}{\nu} \geq \frac{3}{\eta^2}\log\frac{1}{\nu}\left(\ln\log\frac{1}{\nu} + \ln\frac{2}{\zeta}\right)$$

∎

Algorithm 3 Approximate $M_\delta(X)$ with confidence $(1 - \zeta)$

```
function MDelta(X[0,1], δ, ν, η = εδ, ζ)
    N ← (3/η²) log ν⁻¹ (ln(log ν⁻¹) + ln(2ζ⁻¹))
    l ← ν
    h ← 1
    while h − l ≥ ν do
        mid = (h + l)/2
        X_1, ··· , X_N ← samples of X
        p̄ = (1/N)(#X_i, X_i ≥ mid)
        if p̄ ≥ δ then l ← mid
        else h ← mid
    return l
```

4.2 Approximation of $M_\delta(I(.))$ We now apply the binary search for $M_\delta(I(.))$ approximation. Since influence is a discrete random variable bounded by the size n of the graph, we set $\nu = 1/n$, to obtain the following corollary.

COROLLARY 4.1. *Given a graph with n nodes, and seed set S, for given parameters δ, η, and ζ. Using $N \geq \frac{3}{\eta^2}\log n(\ln\log n + \ln(2\zeta^{-1}))$ samples, Algorithm 3 finds an approximation of $M_\delta(I(S))$ with success probability $1 - \zeta$, such that: $M_{\delta+\eta}(I(S)) \leq \widetilde{M}_\delta(I(S)) \leq M_{\delta-\eta}(I(S))$.*

Proof. Direct application of lemma 4.1 with $\nu = 1/n$. ∎

As an example, for the approximation to succeed with high probability, we set $\zeta = 1/n$, and the number of samples required will be $N = O(\log^2 n)$.

Combining the binary search method with the multicriteria approximation, we have the MULTICRITMDELTA as listed in Algorithm 2, with the following approximation guarantee and runtime complexity.

THEOREM 4.1. *Let $\delta \in (0, 1)$ be a constant, and let k be a parameter. Let S_δ^{opt} denote an optimum solution to the* MAXPROBINF *problem for parameter δ, i.e., $M_\delta(I(S^{opt})) \geq M_\delta(I(S'))$, for all $|S'| \leq k$. For any $\epsilon \in (0, 1)$, there exists $N = O(\log^2 n)$ such that, if set S is the solution computed by algorithm* MULTICRITMDELTA *with parameter $\delta' = 2\delta/(1 - \epsilon)$, we have, with high probability, $M_\delta(I(S)) \geq \frac{\delta}{1-\epsilon}M_{2\delta(1+\epsilon)/(1-\epsilon)}(I(S^{opt}_{2\delta/(1-\epsilon)}))$, and*

$|S| = O(k \log n)$. *The running time of* MultiCritMdelta *is* $O(mk \log n(\ln n + \ln \ln n))$.

Proof. Let $\mathcal{A}_{S,\lambda}$ denote the event that the probability $\Pr[I(S) \geq \lambda]$ can be estimated with absolute error $\eta = \delta\epsilon$. The correctness of the algorithm depends on the validation of the feasible λ, which requires all events $\mathcal{A}_{S,\lambda}$ to succeed for all S, λ during the course of the algorithm.

Assume $M = o(n)$, the number of samples taken in the for-loop of Algorithm 2. By the greedy construction of the feasible S, the algorithm needs to check $O(n)$ candidate sets. The number of iterations is the relaxed size of S, $|S| = O(k \ln(M\lambda)) = O(k \ln(n^2))$. Thus the total number of events is: $O(2nk \ln n \log n)$. From equation 4.7, we take a union bound for the failure of all events:

$$2exp(-\frac{\eta^2 M}{3})O(2nk \ln n \log n) \leq \zeta$$

For success w.h.p, we set: $\zeta = 1/n$

$$-\frac{\eta^2 M}{3} \geq \ln n + O(\ln n + \ln \log n)$$
$$\implies M = O(\frac{3}{\eta^2}(\ln n + \ln \log n))$$

We verify that $M = o(n)$ as previously assumed. Thus the total number of samples is: $N = O(\frac{3}{\eta^2} \log n(\ln n + \ln \log n)) = O(\frac{3}{\epsilon^2 \delta^2} \log n(\ln n + \ln \log n))$. The running time is dominated by the complexity of the sampling, which is $Nm = O(mk \log n(\ln n + \ln \log n))$ where m is the number of edges. ∎

5 Empirical Evaluation

We examine the empirical performance of MultiCritMdelta on real world social networks. For a base line method, we also implement ProbInf-Heu, which simply constructs the seed set greedily, using algorithm 3 to maximize $M_\delta(I(.))$ in every step. Also, we compare the quality of the result seed sets with the solutions of MaxExpInf, maximizing influence expectation. Beside the naive direct sampling method of MaxExpInf [13], referred to as NaiveInf, we use DSSA [18], one of the state-of-the-art algorithms, based on reverse sampling [1]. There are other efficient alogrithms, see [1, 4, 21, 20] for example. However, we reemphasize that our problem is different from maximizing the influence expectation, and thus running time comparison is for reference only.

We implemented MultiCritMdelta, ProbInf-Heu, and NaiveInf in C++ with OpenMP. The source code is available at `https://github.com/ngpham/probinf`. For DSSA, we used the implementation made public from [18]. The execution is carried out on a machine with 24 cores and $16GB$ of memory.

We study the following questions. (1) How does $M_\delta(I(S))$ depend on $|S|$ and δ? (2) How are the actual execution times compared to one another and to the theoretical bounds? (3) How does the solution to our algorithms compare with that to the MaxExpInf problem, in terms of both the ExpInf and $M_\delta(I(.))$ objectives?

5.1 Experimental Setup

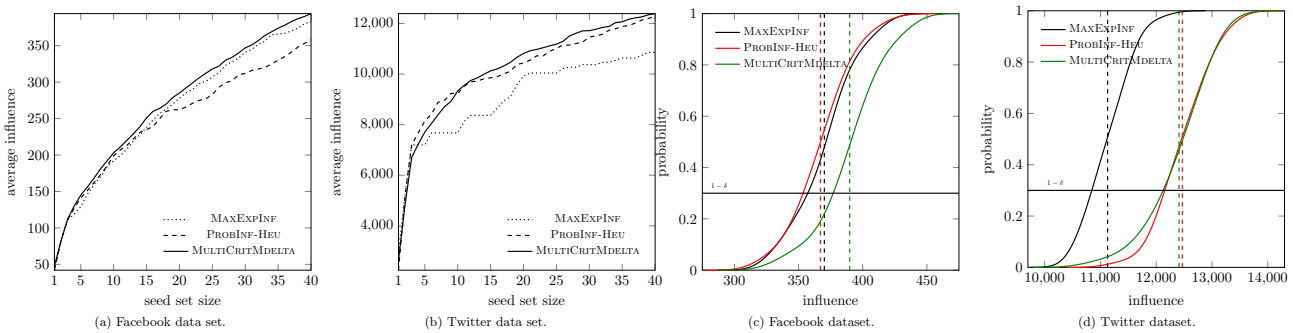
5.1.1 Data sets We use four social network data sets, which were crawled from public sources, as provided by [15]. The basic statistics of the data sets can be found in Table 1.

5.1.2 Parameters of the Algorithms In NaiveInf, we used 10^4 samples, following the claim in [13] that these many samples suffice for datasets of this scale, although the worst case bound is $O(n^2)$ samples. In DSSA, we set the desired multiplicative error to 0.1 (notice that for large input, in scale of millions and over, it is recommended to set this factor to 0.5 [18]).

For our algorithms, ProbInf-Heu and MultiCritMdelta, we take $\eta = 0.1$, which is the additive error for the probability guarantee δ (as discussed in Section 4). Following Theorem 4.1, we choose the number of samples to be $100 \times \log(n)$, where n is the number of nodes of the input graph. Finally, recall that MultiCritMdelta can relax the size of the seed set by a multiplicative factor of $\ln(n)$. In order to make the comparison reasonable, we run it with a smaller budget, so that the final seed set size (after the relaxation) is within the bound of k; this follows the approach in other works which obtain bicriteria approximation results, e.g., [14].

5.1.3 Model parameters. We use the Independent Cascade (IC) model, with the standard setting widely used in the literature, where edge activation probability is set to the reverse of its incident node degree. We also experiment with activation probabilities of $0.01, 0.05, 0.1$, and two random settings, where the activation probability of each edge is picked uniformly randomly in the range $[0.001, 0.05]$ and $[0.001, 0.01]$. This setup is to understand the effect of *sparsity* of the activation.

For ProbInf-Heu and MultiCritMdelta, we conduct two sets of experiments. The first one is configured with probability guarantee δ in $\{0.2, 0.5, 0.7, 0.9\}$. We observe that the resulting seed sets qualities do not vary much with δ (consistent with the observation in [25]), and, therefore, report most of our results for



(a) Facebook data set. (b) Twitter data set. (c) Facebook dataset. (d) Twitter dataset.

Figure 1: Comparison of three algorithms outputs. For ProbInf-Heu and MultiCritMdelta algorithms, the seed sets are optimized with $\delta = 0.7$. Edges activation are random values in the range $[0.001, 0.05]$. Figures (a) and (b) show the average influence versus seed set size. Figures (c) and (d) show the Empirical cumulative distribution function (ECDF) of influence of seed sets of size 40. The vertical color bars indicate the mean values of the corresponding data. The further to the right around probability $= 1 - \delta = 0.3$, the better the influence guarantee.

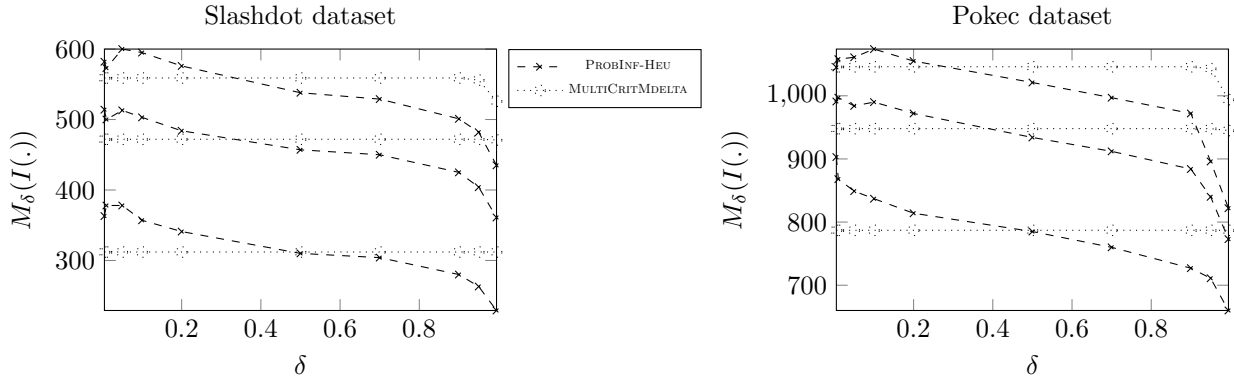


Figure 2: Varying δ, with random edge activation in range $[0.001, 0.01]$. For each algorithm, we report the guarantee influence of the output seed sets of size 15, 30, and 40.

$\delta = 0.7$ because of limited space. In the second set of experiments, we try ProbInf-Heu and MultiCritMdelta with extreme values of δ, which are very close to 0 or 1.

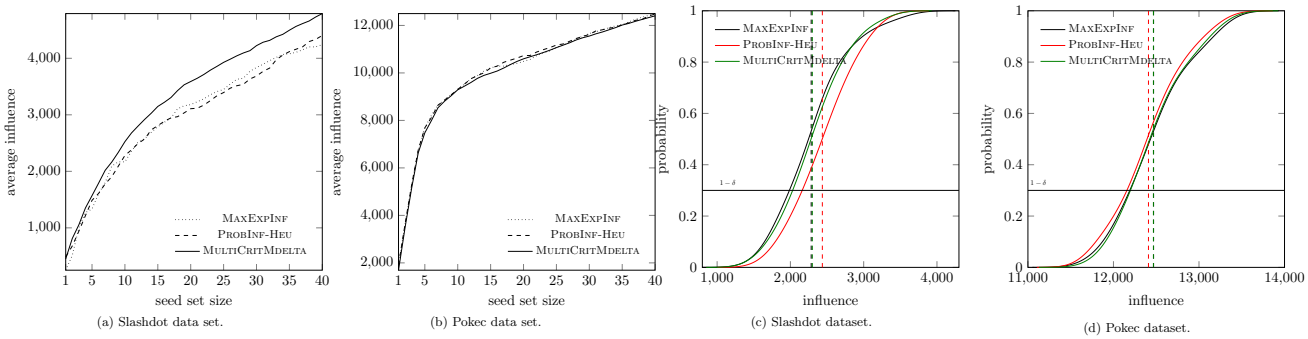
For the constraint on size of the seed set, k, we experiment with all values in $[0, 40]$. It is known from the literature (e.g., [18]) that the influence is quickly saturated when one increases k, such that increasing the seed set does not have significant gain.

Table 1: Data sets.

Data set	Nodes	Edges	Diameter
Facebook	4039	88234	8
Twitter	81306	1768149	7
Slashdot	77360	905468	10
Pokec	1632803	30622564	11

5.2 Results and Evaluation We compare the algorithms by asserting the quality of the output seed sets on the original graph. Recall from section 4.1 that knowing the distribution function would give us complete understanding of $I(.)$, and $M_\delta(I(.))$ can be inferred. Thus, with (fixed) result seed sets, we compute and compare the expected influences, and the empirical cumulative distribution functions (ECDF) [22].

5.2.1 Comparison with MaxExpInf solution. The experiment results with edge activation randomly in the range $[0.001, 0.05]$, are shown in Figure 1. Since DSSA has the same estimation guarantee with NaiveInf, we will refer to these solutions as MaxExpInf. The seed sets are computed by their respective algorithms, where both ProbInf-Heu and MultiCritMdelta has δ set to 0.7. In Figures 1a and 1b, we observe that MultiCritMdelta gives the best seed sets, while ProbInf-Heu gives comparable or bet-



(a) Slashdot data set. (b) Pokec data set. (c) Slashdot dataset. (d) Pokec dataset.

Figure 3: Comparison of three algorithms outputs, complement of Figure 1. For ProbInf-Heu and MultiCritMdelta algorithms, the seed sets are optimized with $\delta = 0.7$. Edges activation are random values in the range $[0.001, 0.05]$. Figures (a) and (b) show the average influence versus seed set size. Figures (c) and (d) show the Empirical cumulative distribution function (ECDF) of influence of seed sets of size 40. The vertical color bars indicate the mean values of the corresponding data. The further to the right around probability $= 1 - \delta = 0.3$, the better the influence guarantee.

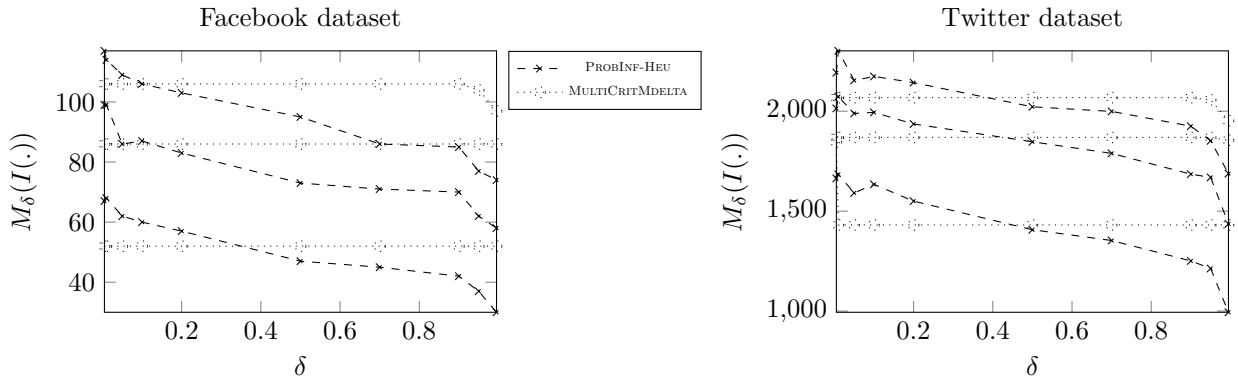


Figure 4: Varying δ, with random edge activation in range $[0.001, 0.01]$. For each algorithm, we report the guarantee influence of the output seed sets of size 15, 30, and 40. This figure complement Figure 2.

ter sets, versus MaxExpInf solutions.

The other experimental results are presented in Figure 3, and Figure 4. Figure 3 shows solutions qualities (the same as Figure 1), using Slashdot and Pokec datasets. We notice that for the Pokec dataset, MaxProbInf solutions do not have significant advantage over MaxExpInf solutions. As per the existential proof of Lemma 2.1, we infer that there could be graphs such that MaxProbInf and MaxExpInf solutions are close to each other. The experimental result on the Pokec dataset implies that this is one such graph. Figure 4 shows the effects of varying δ (similar to Figure 2), using Facebook and Twitter datasets.

5.2.2 Execution time. One of the main advantages of MultiCritMdelta and ProbInf-Heu algorithms is that their execution times are much smaller compared to NaiveInf, while the qualities of the solutions obtained are similar or better. In fact, the MultiCritMdelta algorithm is also faster than ProbInf-Heu. The execution time depends on the number of samples. Each sample complexity depends on the density of the graph, and the edge activation probability. We note that both MultiCritMdelta and ProbInf-Heu algorithms need significantly less number of samples compared to that of the algorithm of [13]. We also observe that MultiCritMdelta asymptotic running time is close to that of DSSA, especially when the activation is *dense*. DSSA has the advantage that it can stop early, by verifying the current estimation error, thus, for network with low activation, it achieves the best runtime. MultiCritMdelta, on the other hand, has to ensure the probabilistic guarantee, and does not check for early termination.

We notice that the variation of δ does not have much affect on the running time, and only report that for $\delta = 0.7$, as shown in Table 2. Also, DSSA runs out of memory on the Pokec dataset, with the most dense

Table 2: Execution Time, for $k = 40$, $\delta = 0.7$ (applied for ProbInf-Heu and MultiCritMdelta), and various edge activation probabilities. The random probability is uniformly in the range $[0.001, 0.05]$. Execution time is measured in seconds.

Dataset	Algorithm	Activation				
		$1/deg$	rand	0.01	0.05	0.1
Facebook	NaiveInf	144	150	138	162	182
	ProbInf-Heu	19	24	22	24	26
	DSSA	3	3	5	5	7
	MultiCritMdelta	8	9	9	9	10
Twitter	NaiveInf	3319	3483	3014	4038	4216
	ProbInf-Heu	662	690	634	799	879
	DSSA	193	216	221	223	230
	MultiCritMdelta	225	242	249	252	260
Slashdot	NaiveInf	1913	1906	1825	2130	2517
	ProbInf-Heu	411	402	383	438	506
	DSSA	83	80	84	92	96
	MultiCritMdelta	227	225	223	232	242
Pokec	NaiveInf	58685	59445	53703	68646	75530
	ProbInf-Heu	19921	20067	16951	19723	22600
	DSSA	5118	5580	5972	6233	—
	MultiCritMdelta	6436	6553	6537	6850	6670

activation setting (0.1). We believe this is due to our hardware limit of 16GB memory.

5.2.3 Probabilistic guarantees We study the probabilistic guarantees of the different algorithms through empirical cumulative distribution functions (ECDF) [22]. Due to space constraint, we only report the results for the case where edge activation are picked randomly in the range $[0.001, 0.05]$, with $\delta = 0.7$. We take the solutions for $k = 40$ and fit their ECDF's by kernel density estimation with Gaussian kernel. These are plotted in figures 1c and 1d, together with vertical lines indicating the respective mean values. The measurement $M_\delta(.)$ can be asserted by the ECDF at probability $(1 - \delta)$. For example, with $\delta = 0.7$, in figure 1d, MultiCritMdelta seed set and ProbInf-Heu seed set is guarantee to have an influence higher than 12100 for 70% of times, which is better than MaxExpInf seed set corresponding value of 10800. Generally, the further the ECDF to the right, around probability $(1 - \delta)$, the higher the quality of the seed set as per $M_\delta(.)$.

5.2.4 Effects of varying δ. This is only applied to ProbInf-Heu and MultiCritMdelta. Figure 2 shows the influence guarantee computed by both algorithms for $\delta \in \{0.005, 0.01, 0.05, 0.1, 0.95, 0.995\}$, with $k \in \{15, 30, 40\}$. We observe that, for a fixed seed set S, the $M_\delta(I(S))$ value from ProbInf-Heu decreases steadily with δ, with sharp decrease when δ is closer to 1. In contrast, the $M_\delta(I(S))$ value from MultiCritMdelta is quite insensitive to δ. Further, ProbInf-Heu has higher $M_\delta(I(S))$ than MultiCritMdelta for small values of δ. In particular, MultiCritMdelta gives good solution in most of the scenarios, except when δ is close to 0 or 1, due to the 1/2 factor in approximating δ.

6 Conclusion

Our results on the MaxProbInf problem suggest that the quality of solutions to influence maximization can be improved by adding probabilistic requirements. We develop the first rigorous approximation algorithms with guarantee bound and efficient running time. Our experimental results show that our algorithms outperform the naive greedy algorithm for the MaxExpInf problem by an order of magnitude, while at the same time there is improvement in the expected influence as well. The results suggest that optimizing based on the MaxProbInf criterion using the MultiCritMdelta algorithm can be a better way to solve the influence maximization problem.

One possible improvement is to incorporate the reverse sampling method [1] and its variation, for example, the early stop detection [18]. It will be good to

quantify which theoretical bounds and guarantees can be achieved for $M_\delta(I(.))$ following that direction.

References

[1] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proc. of 25th ACM-SIAM SODA*, pages 946–957. SIAM, 2014.

[2] Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[3] Wei Chen, Wei Lu, and Ning Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *Proc. AAAI*, 2012.

[4] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. 16th ACM SIGKDD*, pages 1029–1038. ACM, 2010.

[5] Thang N Dinh, Huiyuan Zhang, Dzung T Nguyen, and My T Thai. Cost-effective viral marketing for time-critical campaigns in large-scale social networks. *IEEE/ACM Transactions on Networking (ToN)*, 22(6):2001–2011, 2014.

[6] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*, 2013.

[7] David Easley, Jon Kleinberg, et al. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010.

[8] Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Science, 2005.

[9] Toshihiro Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Trans. on Information and Systems*, 83(3):480–487, 2000.

[10] Sharon Goldberg and Zhenming Liu. The diffusion of networking technologies. In *Proc. 24th ACM-SIAM SODA*, 2013.

[11] Amit Goyal, Francesco Bonchi, Laks VS Lakshmanan, and Suresh Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social network analysis and mining*, 3(2):179–192, 2013.

[12] David Kemp, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. ACM KDD*, pages 137–146, 2003.

[13] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

[14] Andreas Krause, H. Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. *JMLR*, pages 2761–2801, 2008.

[15] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[16] Brendan Lucier, Joel Oren, and Yaron Singer. Influence at scale: Distributed computation of complex contagion in networks. In *Proc. 21th ACM SIGKDD*, 2015.

[17] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[18] Hung T Nguyen, My T Thai, and Thang N Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proc. 2016 SIGMOD*, pages 695–710. ACM, 2016.

[19] Chaitanya Swamy and David B Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *ACM SIGACT News*, 37(1):33–46, 2006.

[20] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proc. 2015 ACM SIGMOD*, 2015.

[21] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. 2014 ACM SIGMOD*, pages 75–86. ACM, 2014.

[22] George R Terrell and David W Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.

[23] Guangmo Tong, Weili Wu, Shaojie Tang, and Ding-Zhu Du. Adaptive influence maximization in dynamic social networks. *IEEE/ACM Transactions on Networking (TON)*, 25(1):112–125, 2017.

[24] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[25] Peng Zhang, Wei Chen, Xiaoming Sun, Yajun Wang, and Jialin Zhang. Minimizing seed set selection with probabilistic coverage guarantee in a social network. In *Proc. 20th ACM KDD*, 2014.

A Proof of Lemma 2.1

Proof. Consider the following graph $G = (V, E)$: we have r cliques $C_1, \ldots, C_r$. Let $|C_i| = C$ for all i. There is a vertex s which has edge (s, v_i) for a specific vertex $v_i \in C_i$. The edges in the cliques C_i all have probability 1. The edges (s, v_i) all have probability $p = 2/r$.

We consider $k = 1$. Then, $E[I(s)] = prC = 2C$. For a vertex $v \in C_i$ for any i, $E[I(v)] \leq C + p^2 rC \leq C + 2pC < 2C$ if $p < 1/2$, which happens if r is large enough. Therefore, the optimum solution to MaxExpInf$(G, 1)$ is $S^* = \{s\}$ and $E[I(S^*)] = 2C$.

Next, consider the MaxProbInf instance $(G, 1, 0.9)$. For $v \in C_i$, $I(v) \geq C$. However, note that $\Pr[I(s) = 1] = (1-p)^r = (1 - \frac{2}{r})^r > 0.1$ for $r \geq 10$. Therefore, for $\delta = 0.9$, $M_\delta(I(S^*)) = 1$. This implies for the instance $(G, 1, 0.9)$, the solution computed using MaxExpInf (ignoring δ) is arbitrarily bad, compared to the optimum solution to the MaxProbInf objective. □ ∎

B Proof of Lemma 2.2

Proof. The proof is by a reduction from MAXIMUM COVERAGE. An instance of this problem is a ground set $U = \{u_1, \ldots, u_n\}$, a collection of subsets $S_1, \ldots, S_m$ of U, and parameter k. The objective is to select a set of k subsets, whose union has the maximum size – this is NP-hard to approximate within a factor of $(1 - 1/e)$.

We construct the same reduction as in [12]. We construct graph $G = (V, E)$ in the following form. We have $V = L \cup R$, where L has m nodes, corresponding to the subsets, and R has n nodes, corresponding to the elements in U. If $u_i \in S_j$, there is a directed edge from the node corresponding to S_j in L to the node corresponding to u_i in R. All edges have diffusion probability 1.

Therefore, for a subset $S \subseteq L$ of seeds, $I(S)$ equals the union of the corresponding subsets. We choose $\delta = 1$. Therefore, there is a set S of size k with $|I(S)| \geq C$ with probability $\delta = 1$ if and only if there is a solution to COVERAGE of size at least C. ∎