Dummy title

Author Name
Dummy University Computer Science Department, USA
http://www.myhomepage.edu
johnqpublic@dummyuni.org

Abstract

The standard 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 which maximizes the influence set size, which can be achieved with a given probability. We show that this objective is not submodular, and design two algorithms for this problem, one of which gives rigorous approximation bounds. We evaluate our algorithms on multiple datasets, and show that they have similar or better performance as the ones optimizing the expected influence, but with additional guarantees on the probability.

2012 ACM Subject Classification Theory of computation → Numeric approximation algorithms

Keywords and phrases Graph data mining, Influence maximization, Probabilistic guarantee

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, let the influence spreads by some probabilistic models (see 2.1 and 3), and take the size of the final influenced set, denoted $I(S)$, as the *influence* of S. One optimization problem, *influence maximization*, is stated as follows: choose 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. 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 the δ - 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.

The first prior work on influence maximization with probabilistic guarantees was by [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. 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.

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 more 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. These were not examined in [25].

2. We develop a multi-criteria approximation algorithm, MultiCritMdelta, with *multiplicative factor* guarantees (instead of both multiplicative and additive guarantee, in [25]). We also design and analyze an efficient sampling method, ProbInf-Heu, 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 IC model. Since it can be shown that $M_\delta(I(S))$ is not submodular (cf. Section 2.5), 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 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 and ProbInf-Heu, have similar or higher (up to 10% in some cases) quantile values than the MaxExpInf solution. More importantly, we observe that the running time of both MultiCritMdelta and ProbInf-Heu is significantly faster (about 10 times faster) compared to the standard algorithm that implements MaxExpInf (due to Kempe et al [13]). 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] which gives a constant-factor approximation to the optimal expected influence.

All omitted proofs and some additional details are presented in the full version of the paper `www.dropbox.com/sh/r6i10b6s18gduap/AAD-4ZUScaq8_-RBeDtCsp2oa?dl=0`.

2 MaxProbInf: Some Basic Results

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 $0 \leq w(u, v) \leq 1$, 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 $w(u, v)$. 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 $wt(v)$ for each node, and $wt(I(S)) = \sum_{v \in A_\tau} wt(v)$ 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:

$$M_\delta(I(S)) = max\{a | \Pr[I(S) \geq a] \geq \delta\} \tag{1}$$

The new optimization problem, referred to as MaxProbInf is defined in the following manner: given an instance $(G = (V, E), k, \delta)$, find a (seed) set S of size k such that $M_\delta(I(S))$ is maximized.

$$\begin{array}{ll} \text{maximize} & M_\delta(I(S)) \\ \text{subject to} & |S| = k. \end{array} \tag{2}$$

The goal of this paper is to study the above optimization problem and give algorithms for it.

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 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.*

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

2.4 Computing $M_\delta(I(S))$ for A Fixed Set S

We show how to compute $M_\delta(I(S))$ for a *fixed* set S. This is a necessary ingredient in computing the solution to the MaxProbInf problem. 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 $M_\delta(I(S))$ efficiently by using Monte-Carlo sampling and binary search; this is crucial in

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

Algorithm 1 ProbInf-Heu - Heuristic Incremental k-influence-set

```
1: function ProbInf-Heu(G(V,E), k, δ, η)
2:     ε ← 1/n
3:     α ← 1/(n log n)
4:     S ← ∅
5:     V' ← V
6:     while |S| < k do
7:         u ← argmax_{v∈V'} MDelta(I(S ∪ {v}) − I(S), δ, η, ε, α)
8:         S ← S ∪ {u}
9:         V' ← V' − {u}
       return S
```

designing our efficient multi-criteria approximation algorithm of Section 5. The sampling algorithm, MDelta(I($\cdot$)), and its proofs are in the full version of the paper; here we just present the final bounds that we need later.

For some random variable X, define $\widetilde{M}_\delta(X, \epsilon, \eta)$ to be an (ϵ, η)-approximation of $M_\delta(X)$ if:

$$\widetilde{M}_\delta(X, \epsilon, \eta) \geq max\{a | \Pr(X \geq a) \geq \delta - \eta\} - \epsilon \tag{3}$$

▶ **Theorem 3** (Sampling Bound for $M_\delta(I(S))$)**.** *On a graph of size n nodes, a given seed set S, for given parameters δ and η there is a Monte-Carlo randomized algorithm that finds an (ϵ, η)-approximation of $M_\delta(I(S))$ with probability of success at least $(1 - \alpha \log(1/\epsilon))$ (for some given parameter α), using $N_{total} \geq \frac{3}{\eta^2} \ln \frac{2}{\alpha} \log \frac{1}{\epsilon}$ samples. In particular, if δ and η are small constants, then only $O((\log n)^2)$ samples are needed to obtain success probability $1 - O\left(\frac{1}{n}\right)$.*

2.5 A Simple Greedy Heuristic for $M_\delta(I(S))$

Motivated by the greedy algorithm of [12] for the MaxExpInf problem, we first consider a greedy algorithm for $M_\delta(I(S))$: start with an empty set S, and proceed in k steps; in each step, select the node $v \notin S$ which yields the highest increment in $M_\delta(I(S))$, and add it to S. The pseudocode is shown in Algorithm 1. The algorithm uses $O(k(\log n)^2)$ samples overall, since it uses k applications of MDelta(I($\cdot$)) and each application uses $O(\log^2 n)$ samples to estimate the influence of the current seed set with high probability (cf. Theorem 6).

Algorithm 1 does not guarantee any approximation bound regarding the influence, since $M_\delta(I(\cdot))$ can be shown to be not a sub-modular set function[25] (a proof can be found in the full paper). However, we find empirically that Algorithm 1 performs quite well on real graph data sets (Section 6).

3 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. These results set a theoretical limit on what can be done to attack the problems.

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. [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 [25]. They propose to measure influence with a probabilistic guarantee; their goal is to find a minimum-sized seed set that achieves a given target influence with a specified probability. They show that their problem is #P-hard, and give an approximation algorithm. The algorithm uses expected influence as the criteria (which requires high number of samples) to select members of the seed set, and then verifies the probability condition. 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 incurs additive error, and does not relies on assumption about the distribution.

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

In this section, we show how to compute $M_\delta(I(S))$ for a *fixed* set S. This is a necessary ingredient in computing the solution to the MaxProbInf problem, which tries to find the S (of given size k) that maximizes $M_\delta(I(S))$. It can be shown 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 $M_\delta(I(S))$

To estimate $M_\delta(I(S))$, it will be useful to consider the following general problem of estimating a random variable that takes values between 0 and 1 (both included).

$$\begin{array}{ll} 0 \le X \le 1 & \text{a random variable,} \\ 0 < \delta \le 1 & \text{a probability threshold,} \\ \text{find} & M_\delta(X) = sup\{a|\Pr[X \ge a] \ge \delta\}. \end{array} \tag{4}$$

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: $M_\delta(X) = sup\{a|\Pr[X \ge a] = \delta\}$, 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 (ϵ,η)-approximation of $M_\delta(X)$. Let $\widetilde{M}_\delta(X,\epsilon,\eta)$ be such an approximation, satisfying:

$$\widetilde{M}_\delta(X,\epsilon,\eta) \ge max\{a|\Pr(X \ge a) \ge \delta - \eta\} - \epsilon \tag{5}$$

The idea is to perform binary search to find the best value of a. 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 smaller than ϵ.

The slack η in the probability comes from the testing for $\Pr(X \geq a) \geq \delta$. For our problem, we will design a Monte-Carlo sampling algorithm. Consider one iteration, with some fixed a, we have to decide: if $\Pr(X \geq a) \geq \delta$, or $\Pr(X \geq a) < \delta$. Take N samples $X_1, X_2, \cdots, X_N$, where N will be specified later. Let $p = \Pr(X \geq a)$ be the unknown, fixed probability. 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 $\bar{p} = \frac{Z}{N}$:

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

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

$$\Pr\left(|\bar{p} - p| \geq \eta\right) \leq 2exp\left(-\frac{\eta^2}{2p+\eta}N\right) \leq 2exp\left(-\frac{\eta^2 N}{3}\right). \tag{6}$$

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 standard method leaves an undecided region when $\bar{p} - \eta < \delta \leq \bar{p} + \eta$. To simplify the decision rule, we allow some slack around δ, and have the following decision rule:

$$\begin{aligned} &\text{If } \bar{p} \geq \delta \implies \text{increase } a; \\ &\text{If } \bar{p} < \delta \implies \text{decrease } a. \end{aligned} \tag{7}$$

Algorithm 2 shows the pseudo code for the above algorithm. Next, we will show that it gives an (correct) (ϵ, η)-approximation of $M_\delta(X)$ (as defined in 5). Then we will analyze the required number of samples to ensure correctness with high probability.

▶ **Lemma 4.** *Algorithm 2 finds an (ϵ, η)-approximation of $M_\delta(X)$, assuming that all the Monte-Carlo tests are successful.*

Proof. It follows directly from the binary search that the final value a is within $\pm\epsilon$ of the maximum of a. The decision rules in Equation (7) is equivalent to: increase a when $p \geq \delta - \eta$, decrease a when $p < \delta + \eta$. This holds for every iteration, thus, $\Pr(X \geq a)$ is guaranteed to be at least $\delta - \eta$. □ ◀

In our algorithm, N is the number of samples in each iteration. Suppose we want to compute an (ϵ, η)-approximation of $M_\delta(X)$ with high (confidence) probability, we use the following lemma:

▶ **Lemma 5.** *Algorithm 2 finds an (ϵ, η)-approximation of $M_\delta(X)$, with probability of success at least $(1 - \alpha\log(1/\epsilon))$ (for some given parameter α), using $N_{total} \geq \frac{3}{\eta^2}\ln\frac{2}{\alpha}\log\frac{1}{\epsilon}$ samples.*

Proof. For each iteration to be successful with probability at least $(1 - \alpha)$, we bound the number of samples (in one iteration) by Chernoff bound:

$$2exp\left(-\frac{N\eta^2}{3}\right) \leq \alpha \iff N \geq \frac{3}{\eta^2}\ln\frac{2}{\alpha}. \tag{8}$$

The number of iterations is: $\log(1/\epsilon)$. Using union bound, we have the success probability for the entire algorithm: $(1 - \alpha \log(1/\epsilon))$. Hence, given parameters $\delta, \epsilon, \eta, \alpha$, Algorithm 2 finds $\widetilde{M}_\delta(X, \epsilon, \eta)$ with success probability at least $(1 - \alpha \log(1/\epsilon))$, and requires at least $N_{total} = \frac{3}{\eta^2} \ln \frac{2}{\alpha} \log \frac{1}{\epsilon}$ samples. □ ◀

One may ask why Lemma 5 uses confidence of $(1 - \alpha \log(1/\epsilon))$, instead of the canonical form $(1 - \zeta)$ where $\zeta < 0$. Indeed, we choose the representation of $\alpha \log(1/\epsilon)$ to facilitate the analysis when applied to the graph influence problem, which is presented next.

Algorithm 2 Approximate $M_\delta(X)$ with confidence $(1 - \alpha \log(1/\epsilon))$

```
1: function MDELTA(X[0,1], δ, ϵ, η, α)
2:     N ← (3/η²) ln(2/α)
3:     l ← ϵ
4:     h ← 1
5:     while h − l ≥ ϵ do
6:         mid = (h + l)/2
7:         X_1, ··· , X_N ← samples of X
8:         p̄ = (1/N)(#X_i, X_i ≥ mid)
9:         if p̄ ≥ δ then l ← mid
10:        else h ← mid
       return l
```

▶ **Theorem 6.** *On a graph of size n nodes, a given seed set S, for given parameters δ and η (assumed to be small constants) Algorithm 2 finds an (ϵ, η)-approximation of $M_\delta(I(S))$ using $O((\log n)^2)$ samples with success probability $1 - \Omega\left(\frac{1}{n}\right)$.*

Proof. Consider the choice of parameters when applying Algorithm 2 to estimate $I(S)$. Because $1 \leq I(S) \leq n$, we normalize $I(S)$ into the range $(0, 1]$, and chose $\epsilon = 1/n$. For the algorithm to succeed with probability at least $(1 - 1/n)$, requires $\alpha = \frac{1}{n \log n}$ (by Lemma 5). Hence the number of samples is: $\frac{3}{\eta^2} \log n \ln \frac{n \log n}{2} = O(\log^2 n)$. □ ◀

5 A Multi-criteria Approximation Algorithm for Computing MaxProbInf

Motivated by the hardness from Lemma 2, and the non-submodularity of $M_\delta(I(S))$, 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 $M_\delta(I(S))$ problem: find S such that $M_\delta(I(S)) \geq \gamma a$ and $|S| \leq \beta k$, where a corresponds to the influence size for the optimal solution, and $\gamma < 1, \beta > 1$ are the relaxation parameters. The quantity a will be determined through the binary search.

We use ideas from the result of Krause et al. [14] on minimum cost submodular cover problem; our problem has crucial differences, because of which we cannot directly use their algorithm. Assume we have N samples $G_1, \ldots, G_N$, where $G_i = (V, E_i)$ is the ith sample, obtained by the *triggering set* technique (we discuss bounds on N later in Theorem 10). Let $F_i(S)$ denote the total influence due to S in sample i – this equals the sum of the component sizes containing S. Define $F_{i,\lambda}(S) = \min\{F_i(S), \lambda\}$.

The probabilistic guarantee in $M_\delta(I(S))$ requires finding S that ensures that $|\{i : F_{i,\lambda}(S) \geq \lambda\}| \geq \delta N$. However, it is quite challenging to find a set S with such a property in an incremental manner: a node v might increase $F_{i,\lambda}(S)$ only in some samples i. Thus,

Algorithm 3 k-Influence set by MultiCritMdelta

```
1: function MultiCritMdelta(G(V, E), k, δ, n, N)
2:     l ← 1
3:     h ← n
4:     for step ← [1 ... log n] do
5:         S ← ∅
6:         λ ← (l + h)/2
7:         C ← δλ, and F(S) ← F_λ(S)
8:         while F(S) < C do
9:             Pick node j that minimizes w_j / (F(S∪{j}) − F(S))
10:            S ← S ∪ {j}
11:        feasible if |S| ≤ k ln(Nλ) : l ← λ
12:        infeasible if |S| > k ln(Nλ): h ← λ
       return S
```

it is not clear how to pick nodes each time that ensure influence in many samples is high, eventually. Instead, we will consider the average defined as

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

A critical observation is that $F_{i,\lambda}(\cdot)$ and thus $F_\lambda(\cdot)$ are submodular [9]. Hence, greedy approach to maximize $F_\lambda(\cdot)$ will give a guaranteed approximation. We show that maximizing $F_\lambda(S)$ is good enough due to the following property.

▶ **Lemma 7.** *If $|\{i$ such that $F_{i,\lambda}(S) \geq \lambda\}| \geq \delta N$, then $F_\lambda(S) \geq \delta\lambda$. On the other hand, if $F_\lambda(S) \geq \delta\lambda$, then $|\{i$ such that $F_{i,\lambda}(S) \geq \delta\lambda/2\}| \geq \delta N/2$.*

Lemma 7 motivates the following strategy to solve the $M_\delta(I(S))$ problem, which is a variant of the minimum cost submodular cover problem [8]: find S such that $F_\lambda(S) \geq \delta\lambda$ and $\text{cost}(S) = \sum_{j \in S} w_j$ is minimized. Note that the standard submodular cover problem requires $F(S) = F(V)$, where V is the ground set.

Fast implementation of algorithm MultiCritMdelta. The pseudocode is shown in Algorithm 3. It uses 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 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. Theorem 8 shows the approximation guarantee of this algorithm.

Let $S^*, \lambda^* = \text{argmax}_{S,\lambda} |\{i \text{ such that } F_{i,\lambda}(S) \geq \lambda\}| \geq \delta N$, where the maximization is over sets S of size at most k.

▶ **Theorem 8.** *Let S^* be an optimum solution, as defined above. The set S computed by Algorithm 3 satisfies: (1) $F_\lambda(S) \geq \lambda^*\delta$ and (2) $|S| = O(\ln N\lambda)k$, where $k = |S^*|$.*

Our proof of Theorem 8 is a variation of the proof by Wolsey [24], but has a crucial difference from minimum submodular cover, in that $F(S)$ need not equal to $F(V)$. 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 9.** *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

$$\begin{aligned} \text{(D) max} \quad & \sum_S (C - F(S)) y_S \quad \text{such that} \\ \sum_{S: j \notin S} \rho_j(S) y_S &\leq w_j \text{ for all } j \\ y_S &\geq 0 \end{aligned}$$

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 Theorem 8. □

Finally, we combine this with the right number of samples (applying Theorem 6 using $\Theta(n)$ samples), and put everything together to obtain the following theorem.

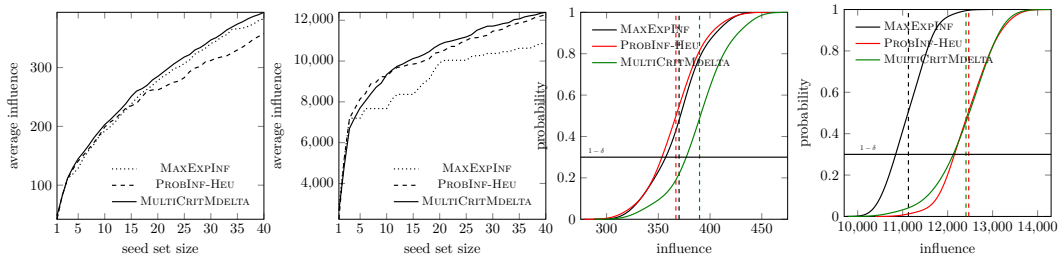
▶ **Theorem 10.** *Let S^* denote the optimum solution to the* MaxProbInf *problem. Let $N = \Theta(\frac{c}{\epsilon^2 \delta} n)$ samples, for a constant c. Then,* MultiCritMdelta *gives a solution S such that $M_{\delta/2}(I(S)) \geq \frac{(1-\epsilon)\delta}{2} M_\delta(I(S^*))$.*

The factor $1/2$ for δ in the approximation solution comes from Lemma 7. Note that we need $\Theta(n)$ samples so that all subsets (there can be exponential number of them) are accurately sampled with high probability by a union bound (applying Theorem 6). However, our experimental results show that, in practice, we need substantially less number of samples — $\Theta(\log n)$ – to guarantee a good approximation.

6 Empirical Evaluation

We examine the empirical performance of our algorithms (ProbInf-Heu and MultiCritMdelta) on real world social networks. We study the following questions.

- How does $M_\delta(I(S))$ depend on $|S|$ and δ?
- How are the actual execution times compared to one another and to the theoretical sampling sizes?
- How does the performance of our algorithms compare with the solution to the MaxExpInf problem, in terms of both the ExpInf and $M_\delta(I(S))$ objectives?



(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.

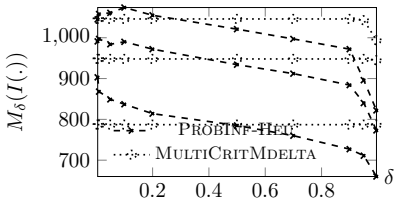


Figure 2 Pokec Dataset: Varying δ, with low edge activation in range $[0.001, 0.01]$. For each algorithm, we report the guarantee influence of the output seed sets of size 15, 30, and 40.

6.1 Experimental Setup

Data sets and tested algorithms. 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.

We implement our algorithms for MaxProbInf (ProbInf-Heu, and MultiCritMdelta), and compare them with the greedy algorithm for MaxExpInf (as described in [13]). There have been more recent advances on algorithms to scale the implementation of influence maximization, e.g., [1, 4, 21, 20]; however, these are pretty complicated, and so we use the algorithm of [13] for comparison. Further, we use 10^4 samples, following the claim in [13] that these many samples suffice for datasets of this scale, though the worst case bound is $O(n^2)$ samples,

For 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 6, 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].

Our codes are implemented in C++ with OpenMP, and executed on a server with 24 cores and $16GB$ of memory. The source code is published on github.

Model parameters. We use the IC model with activation probabilities of $0.01, 0.05, 0.1$, and a two random settings, where the activation probability of each edge is picked uniformly randomly in the range $[0.001, 0.05]$ and $[0.001, 0.01]$.

For ProbInf-Heu and MultiCritMdelta, we conduct two sets of experiments. The

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

first one is configured with probability guarantee δ in $\{0.2, 0.5, 0.70.9\}$. We observe that the resulted seed sets qualities do not vary much with δ (consistent with the observation in [25]), and, therefore, report most of our results for $\delta = 0.7$ because of the 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 related works that the influence is quickly saturated when one increases k, such that increasing the seed set does not have significant gain.

6.2 Results and Evaluation

We compare the algorithms by asserting the quality of the returned seed sets on the original graph, in term of average influence, and empirical cumulative distribution functions (ECDF).

Comparison with ExpInf solution. The experiment results with edge activation randomly in the range $[0.001, 0.05]$, are shown in Figure 1. 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 better sets, versus MaxExpInf solutions.

Execution time. One of the main advantages of MultiCritMdelta and ProbInf-Heu algorithms is that their execution times are much smaller compared to the greedy algorithm for MaxExpInf(as described in [13]), while the qualities of the solutions obtained are similar or better. In fact, the MultiCritMdelta algorithm is the fastest among the three. 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 notice that the variation of δ do not have much affect on the runtime, and only report the runtime for $\delta = 0.7$, as shown in Table 2.

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 $M_\delta(.)$.

Table 2 Execution Time, for $k = 40$, $\delta = 0.7$ (where applicable), and various edge activation probabilities. The random probability is uniformly in the range $[0.001, 0.05]$. Runtime is measured in seconds. mexp, mprob, and mcrit indicates MaxExpInf, ProbInf-Heu, and MultiCritMdelta respectively.

Data set	rand			0.01			0.05			0.1		
	mexp	mprob	mcrit	mexp	mprob	mcrit	mexp	mprob	mcrit	mexp	mprob	mcrit
Facebook	150	24	**9**	138	22	**9**	162	24	**9**	182	26	**10**
Twitter	3483	690	**242**	3014	634	**249**	4038	799	**252**	4216	879	**260**
Slashdot	1906	402	**225**	1825	383	**223**	2130	438	**232**	2517	506	**242**
Pokec	59445	20067	**6553**	53703	16951	**6537**	68646	19723	**6850**	75530	22600	**6670**

Effects of varying δ. 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 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.

References

1 Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proc. of 25th ACM-SIAM SDA*, 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 D. Easley and J. Kleinberg. *Networks, Crowds and Markets: Reasoning About A Highly Connected World*. Cambridge University Press, New York, NY, 2010.

8 S. 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 SDA*, 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 D. Kempe, J. Kleinberg, and E. 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. URL: `http://www.theoryofcomputing.org/articles/v011a004`, `doi:10.4086/toc.2015.v011a004`.

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. URL: `http://doi.acm.org/10.1145/2783258.2783334`, `doi:10.1145/2783258.2783334`.

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 ICMD*, pages 695–710. ACM, 2016.

19 C. Swamy and D. Shmoys. Algorithms column: Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, 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 L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, page 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.