Why approximate when you can get the exact? Optimal Targeted Viral Marketing at Scale

Xiang Li, J. David Smith
CISE Department
University of Florida
Gainesville, FL, 32611
Email: {xixiang, jdsmith}@cise.ufl.edu

Thang N. Dinh
CS Department
Virginia Commonwealth University
Richmond, VA 23284
Email: tndinh@vcu.edu

My T. Thai
CISE Department
University of Florida
Gainesville, FL, 32611
Email: mythai@ufl.edu

Abstract—One of the most central problems in viral marketing is Influence Maximization (IM), which finds a set of k seed users who can influence the maximum number of users in online social networks. Unfortunately, all existing algorithms to IM, including the state of the art SSA and IMM, have an approximation ratio of $(1-1/e-\epsilon)$. Recently, a generalization of IM, Cost-aware Target Viral Marketing (CTVM), asks for the most cost-effective users to influence the most relevant users, has been introduced. The current best algorithm for CTVM has an approximation ratio of $(1-1/\sqrt{e}-\epsilon)$.

In this paper, we study the CTVM problem, aiming to optimally solve the problem. We first highlight that using a traditional two stage stochastic programming to exactly solve CTVM is not possible because of scalability. We then propose an almost exact algorithm TIPTOP, which has an approximation ratio of $(1-\epsilon)$. This result significantly improves the current best solutions to both IM and CTVM. At the heart of TIPTOP lies an innovative technique that reduces the number of samples as much as possible. This allows us to exactly solve CTVM on a much smaller space of generated samples using Integer Programming. While obtaining an almost exact solution, TIPTOP is very scalable, running on billion-scale networks such as Twitter under three hours. Furthermore, TIPTOP lends a tool for researchers to benchmark their solutions against the optimal one in large-scale networks, which is currently not available.

Keywords—*Viral Marketing; Influence Maximization; Algorithms; Online Social Networks; Optimization*

I. INTRODUCTION

With billions of active users, Online Social Networks (OSNs) have been one of the most effective platforms for marketing and advertising. Through "word-of-mouth" exchanges, so-called viral marketing, the influence and product adoption can spread from few key users to billions of users in the network. To identify those key users, a great amount of work has been devoted to the Influence Maximization (IM) problem which asks for a set of k seed users that maximize the expected number of influenced nodes. Taking into account both arbitrary cost for selecting a node and arbitrary benefit for influencing a node, a generalized problem, Cost-aware Targeted Viral Marketing (CTVM), has been introduced recently [1]. Given a budget B, the goal is to find a seed set S with the total cost at most B that maximizes the expected total benefit over the influenced nodes.

Solutions to both IM and CTVM either suffer from scalability issues or limited performance guarantees. Since IM is NP-hard, Kempe et al. [2] introduced the first $(1-\frac{1}{e}-\epsilon)-approximation$ $algorithm$ for any $\epsilon > 0$. However, it is not scalable. This work has motivated a vast amount of work on IM

in the past decade [3]–[10] (and references therein), focusing on solving the scalability issue. The most recent noticeable results are IMM [5] and SSA [6] algorithms, both of which can run on large-scale networks with the same performance guarantee of $(1-\frac{1}{e}-\epsilon)$. Meanwhile, BCT was introduced to solve CTVM with a ratio of $(1-1/\sqrt{e}-\epsilon)$ [1].

Despite a great amount of aforementioned works, none of these attempts to solve IM or CTVM *exactly*. The lack of such a solution makes it impossible to evaluate the practical performance of existing algorithms in real-world datasets. That is: how far from optimal are the solutions given by existing algorithms on billion-scale OSNs, in addition to their theoretical performance guarantees? This question has remained unanswered til now.

Unfortunately, obtaining exact solutions to CTVM (and thus to IM) is very challenging as the running time is no longer in polynomial time. Due to the nature of the problem, stochastic programming (SP) is a viable approach for optimization under uncertainty when the probability distribution governs the data is given [11]. However traditional SP-based solutions to various problems in NP-hard class are only for small networks with a few hundreds nodes [11]. Thus directly applying existing techniques to IM and CTVM is not suitable as OSNs consist of millions of users and billions of edges. Furthermore, the theory developed to assess the solution quality such as those in [11], [12], and the references therein, only provide approximate confidence interval. Therefore, sufficiently large samples are needed to justify the quality assessment. This requires us to develop novel stochastic programming techniques to exactly solve CTVM on large-scale networks.

In this paper, we provide the very first (almost) exact solutions to CTVM. To tackle the above challenges, we develop two innovative techniques: 1) Reduce the number of samples as much as possible so that the SP can be solved in a very short time. This requires us to tightly bound the number of samples needed to generate a candidate solution. 2) Develop novel computational method to assess the solution quality with just enough samples, where the quality requirement is given a priori. These results break the tradition and cross the barriers in stochastic programming theory where solving SP on large-scale networks had been thought impractical. Our contributions are summarized as follows:

- Design an (almost) exact solution to CTVM using two-stage stochastic programming (T-EXACT) which utilizes the sample average approximation method to reduce the number of realizations. Using T-EXACT, we illustrate that traditional stochastic programming techniques badly suffer the scalability issue.

- Develop a scalable and exact algorithm to CTVM with an approximation ratio of $(1 - \epsilon)$: The Tiny Integer Program with Theoretically OPtimal results (TIPTOP). Being able to obtain the near exact solution, TIPTOP is used as a benchmark to evaluate the absolute performance of existing solutions in billion-scale OSNs.

- Conduct extensive experiments confirming that the theoretical performance of TIPTOP is attained in practice. That said, it is feasible to compute 98% optimal solutions to CTVM on billion-scale OSNs in under three hours. The experiments further confirm that our sampling reductions are significant in practice – with reductions by a factor of 10^3 on average.

Related works. Kempe et al. [2] is the first to formulate viral marketing as the IM optimization problem. They showed the problem to be NP-complete and devised an $(1 - 1/e - \epsilon)$ approximation algorithm on two fundamental cascade models, Linear Threshold (LT) and Independent Cascade (IC). In addition, IM cannot be approximated within a factor $(1 - \frac{1}{e} + \epsilon)$ [13] under a typical complexity assumption. Computing the exact influence is shown to be #P-hard [4]. Following [2], a series of work have been proposed, focused on improving time complexity. A notable one is Leskovec et al [3], in which the lazy-forward heuristic (CELF) was used to obtain about 700-fold speed up. The major bottle-neck in these works is the inefficiency in estimating the influence spread. The diffusion process also used to restrain propagation in social networks [14].

Recently, Borgs et al. [15] have made a theoretical breakthrough and introduced a novel sampling approach, called *Reverse Influence Sampling* (RIS), which is a foundation for the later works. Briefly, RIS captures the influence landscape of $G = (V, E, p)$ through generating a hypergraph $\mathcal{H} = (V, \{\mathcal{E}_1, \mathcal{E}_2, \ldots\})$. Each hyperedge $\mathcal{E}_j \in \mathcal{H}$ is a subset of nodes in V and constructed as follows: 1) selecting a random node $v \in V$ 2) generating a sample graph $g \sqsubseteq G$ and 3) returning $\mathcal{E}_j$ as the set of nodes that can reach v in g. Observe that $\mathcal{E}_j$ contains the nodes that can influence its source v. If we generate multiple random hyperedges, influential nodes will likely appear more often in the hyperedges. Thus a seed set S that *covers* most of the hyperedges will likely maximize the influence spread. Here a seed set S covers a hyperedge $\mathcal{E}_j$, if $S \cap \mathcal{E}_j \neq \emptyset$. Therefore, IM can be solved using the following framework. 1) Generate multiple random hyperedges from G. 2) Use the greedy algorithm for the Max-coverage problem [16] to find a seed set S that covers the maximum number of hyperedges and return S as the solution. The core issue in applying the above framework is that: *How many hyperedges are sufficient to provide a good approximation solution?*

Based on RIS, Borgs et al. [15] presented an $O(kl^2(m + n) \log^2 n/\epsilon^3)$ time algorithm for IM under IC model. It returns a $(1 - 1/e - \epsilon)$-approximate ratio with probability at least $1 - n^{-l}$. In practice, the proposed algorithm is, however, less than satisfactory due to the rather large hidden constants. In a sequential work, Tang et al. [5] reduce the running time to $O((k + l)(m + n) \log n/\epsilon^2)$ and show that their algorithm is efficient in billion-scale networks. However, the number of generated samples can be arbitrarily large. The state of the art SSA algorithm is introduced in [6] that outperforms existing methods several orders of magnitudes w.r.t running time. The algorithm keeps generating samples and stops at exponential check points to verify (stare) if there is adequate statistical evidence on the solution quality for termination. However, SSA

does not put an effort in minimizing the number of samples at the stopping point to verify the candidate solution, which is needed to solve the Integer Programming (IP). All of these works have a $(1 - 1/e - \epsilon)$-approximation ratio.

In generalizing IM, Nguyen and Zheng [17] investigated the BIM problem in which each node can have an arbitrary selecting cost. They proposed a $(1 - 1/\sqrt{e} - \epsilon)$ approximation algorithm (called BIM) and two heuristics. However, none of the proposed algorithms can handle billion-scale networks. Recently, Nguyen *et. al* introduced CTVM and presented a scalable $(1 - 1/\sqrt{e} - \epsilon)$ algorithm [1]. They also showed that straightforward adaption of the methods in [5], [15] for CTVM can incur an excessive number of samples, thus, are not efficient enough for large networks.

Organization. The rest of the paper is organized as follows. In Section II, we present the network model, propagation models, and the problem definition. Section III presents our T-EXACT algorithm for CTVM. Our main contribution, TIPTOP is introduced in Section IV. We analyze the performance of TIPTOP in Section V. Experimental results on real social networks are shown in Section VI. And finally Section VII concludes our paper.

II. MODELS AND PROBLEM DEFINITIONS

Let $G = (V, E, c, b, p)$ be a social network with a node set V and a directed edge set E, with $|V| = n$ and $|E| = m$. Each node $u \in V$ has a selecting cost $c(u) \geq 0$ and a benefit $b(u) \geq 0$ if u is influenced[1]. Each directed edge $(u, v) \in E$ is associated with an influence probability $p_{uv} \in [0, 1]$.

Given G and a subset $S \subset V$, referred to as the *seed set*, in the IC model the influence cascades in G as follows. The influence propagation happens in round $t = 1, 2, 3, \ldots$. At round 1, nodes in the seed set S are *activated* and the other nodes are *inactive*. The cost of activating the seed set S is given $c(S) = \sum_{u \in S} c(u)$. At round $t > 1$, each newly activated node u will independently activate its neighbor v with a probability p_{uv}. Once a node becomes activated, it remains activated in all subsequent rounds. The influence propagation stops when no more nodes are activated.

Denote by $\mathbb{I}(S)$ the expected number of activated nodes given the seed set S. We call $\mathbb{I}(S)$ the *influence spread* of S in G under the IC model.

Mathematically, we view the probabilistic graph $\mathcal{G}$ as a generative model for deterministic graphs. A deterministic graph $g = (V, E_s)$ is generated from $\mathcal{G}$ by selecting each edge $(u, v) \in E$, independently, with probability p_{uv}. We refer to g as a *realization* or a *sample* of G and write $g \sqsubseteq \mathcal{G}$. The probability that g is generated from G is

$$\Pr[g] = \prod_{e \in E_s} p_e \prod_{e \in E \setminus E_s} (1 - p_e).$$

Let $m = |E|$, there are $W = 2^m$ possible realizations of $\mathcal{G}$. We number those realizations as $G^1 = (V, E^1), G^2 = (V, E^2), \ldots, G^W = (V, E^W)$, where $E^1, E^2, \ldots, E^W$ are all possible subsets of E. The influence spread of a seed set S equals the expected number of nodes reachable from S over all possible sample graphs, i.e.,

---

[1]The cost of node u, $c(u)$, can be estimated proportionally to the centrality of u (how important the respective person is), e.g., out-degree of u [17]. Additionally, the node benefit $b(u)$ refers to the gain of influencing node u, e.g., 1 for each node in our targeted group and 0 outside.

$$\mathbb{I}(S) = \sum_{g \sqsubseteq G} \Pr[g]|R(g,S)|, \tag{1}$$

where $\sqsubseteq$ denotes that the sample graph g is generated from G with a probability denoted by $\Pr[g]$, and $R(g,S)$ denotes the set of nodes reachable from S in g.

Similarly, the *expected benefit* of a seed set S is defined as the expected total benefit over all influenced nodes, i.e.,

$$\mathbb{B}(S) = \sum_{g \sqsubseteq G} \Pr[g] \sum_{u \in R(g,S)} b(u). \tag{2}$$

We are now ready to define the CTVM problem as follows.

Definition 1 (Cost-aware Targeted Viral Marketing - CTVM). *Given a graph $G = (V, E, c, b, p)$ and a budget $BD > 0$, find a seed set $S \subset V$ with the total cost $c(S) \leq BD$ to maximize the expected benefit $\mathbb{B}(S)$.*

III. EXACT ALGORITHM VIA STOCHASTIC PROGRAMMING

In this section, we first present our T-EXACT (traditional exact) algorithm based on two-stage SP. We then discuss the scalability issue of traditional SP which is later verified in our experiment, shown in Section VI.

A. Two-stage Stochastic Linear Program

Given an instance $G = (V, E, c, b, p)$ of CTVM, we first use integer variables s_v to represent whether or not node v is selected as a seed node, i.e., $s_v = 1$ if node v is selected (aka $v \in S$ where S denotes the seed set) and 0, otherwise. Here $n = |V|$ is the number of nodes and we assume nodes are numbered from 1 to n. We impose on s the budget constraint $\sum_{v \in V} s_v c_v \leq BD$. Variables s are known as first stage variables. The values of s are to be decided before the actual realization of the uncertain parameters in G.

We associate with each node pair (u, v) a random Bernoulli variable ξ_{uv} satisfying $\Pr[\xi_{uv} = 1] = p_{uv}$ and $\Pr[\xi_{uv} = 0] = 1 - p_{uv}$. For each realization of $\mathcal{G}$, the values of ξ_{uv} are revealed to be either 0 or 1 and we can compute the benefit of the seed set S implied by s. To do so, we define integer variable x_v to be the state of node v when the propagation stops, i.e., $x_v = 1$ if v is eventually activated and 0, otherwise.

The benefit obtained by seed set can be computed using a second stage integer programming, denoted by $B(s, x, \xi)$ as follows.

$$B(s, x, \xi) = \max \sum_{v \in V} b(v) x_v \tag{3}$$

$$\text{s. t.} \sum_{u \in IR(\xi,v)} s_u \geq x_v, \ v \in V, \tag{4}$$

$$s_u \in \{0, 1\}, x_v \in \{0, 1\} \tag{5}$$

where $IR(\xi, v)$ denotes the set of nodes u so that there exists a path from u to v given the realization ξ.

The two-stage stochastic linear formulation for the CTVM problem is as follows.

$$\max_{s \in \{0,1\}^n} \mathbb{E}[B(s, x, \xi)] \tag{6}$$

$$\text{s. t.} \sum_{v \in V} s_v c_v \leq BD \tag{7}$$

$$\text{where } B(s, x, \xi) \text{ is given in (3)-(5)} \tag{8}$$

The objective is to maximize the expected benefit of the activated nodes $\mathbb{E}[B(s, x, \xi)]$, where $B(s, x, \xi)$ is the optimal value of the second-stage problem. This stochastic programming problem is, however, not yet ready to be solved with a linear algebra solver.

1) Discretization: To solve a two-stage stochastic problem, one often need to discretize the problem into a single (very large) linear programming problem. That is we need to consider all possible realizations $G^l \sqsubseteq \mathcal{G}$ and their probability masses $\Pr[G^l]$. Denote by $\{\xi^l\}_{ij}$ the adjacency matrix of the realization $G^l = (V, E^l)$, i.e., $\xi_{ij}^l = 1$, if $(i, j) \in E^l$, and 0, otherwise. Since the objective involves only the *expected cost* of the *second stage* variables x_{ij}, the two-stage stochastic program can be discretized into a mixed integer programming, denoted by MIP_F as follows.

$$\max \sum_{l=1}^{W} \Pr[G^l] \sum_v b(v) x_v^l \tag{9}$$

$$\text{s. t.} \sum_{v \in V} s_v c_v \leq BD \tag{10}$$

$$\sum_{u \in IR(G^l, v)} s_u \geq x_v^l, \ v \in V, G^l \sqsubseteq \mathcal{G} \tag{11}$$

$$s_u \in \{0, 1\}, x_v^l \in \{0, 1\} \tag{12}$$

Lemma 1. *The expected number of non-zeros in T-EXACT is $T \sum_{u \in V} I(u)$ where T is the number of graph realizations and $I(u)$ denotes the expected influence of node u.*

Proof: For each node $u \in V$, the number of constraints (11) that u was on the left hand sides equal the number of nodes that u can reach to. The mean number of constraints (11) that u participates into is, hence, $I(u)$. Taking the sum over all possible $u \in V$, we have the expected size of the T-EXACT with T graph realizations is $T \times \sum_{u \in V} I(u)$. ∎

2) Realization Reduction: An approach to reduce the number of realizations in T-EXACT is to apply the Sample Average Approximation (SAA) method. We generate independently T samples $\xi^1, \xi^2, \cdots, \xi^T$ using Monte Carlo simulation (i.e. to generate each edge $(u, v) \in E$ with probability p_{uv}). The expectation objective $q(s) = \mathbb{E}[B(s, x, \xi)]$ is then approximated by the sample average $\hat{q}_T(x) = \frac{1}{T} \sum_{l=1}^{T} \sum_v (b(v) x_v^l)$, and the new formulation is then

$$\max \quad \frac{1}{T} \sum_{l=1}^{T} \sum_v b(v) x_v^l$$
$$\text{s. t.} \quad \text{Constraints}(10) - (12)$$

Under some regularity conditions $\frac{1}{T} \sum_{j=1}^{T} \sum_{i<j} (1 - x_{ij}^l)$ converges pointwise with probability 1 to $\mathbb{E}[B(s, x, \xi)]$ as $T \to \infty$. Moreover, an optimal solution of the sample average approximation provides an optimal solution of the stochastic programming with probability approaching one exponentially fast w.r.t. T. Formally, denote by s^* and $\hat{s}$ the optimal solution of the stochastic programming and the sample average approximation, respectively. For any $\epsilon > 0$, it can be derived from Proposition 2.2 in [18] that

$$\Pr[\mathbb{E}[B(s, \hat{x}, \xi)] - \mathbb{E}[B(s, x^*, \xi)] > \epsilon]$$
$$\leq \exp\left(-T \frac{\epsilon^2}{n^4} + k \log n\right) \tag{13}$$

Equivalently, if $T \geq \frac{n^4}{\epsilon^2}(k \log n - \log \alpha)$, then $\Pr[\mathbb{E}[B(s, \hat{x}, \xi)] - \mathbb{E}[B(s, x^*, \xi)] < \epsilon] > 1 - \alpha$ for any $\alpha \in (0, 1)$.

B. Scalability Issues of Traditional Stochastic Optimization

While T-EXACT was designed based on a standard method for stochastic programming, traditional methods can only be applied for small networks, up to few hundreds nodes [11].

There are three major scalability issues when applying SAA and using T-EXACT for the influence maximization problem. First, the samples have a large size $O(m)$, where $m = |E|$ is the number of edges in the network. For large networks, m could be of size million or billion. As a consequence, we can only have a small number of samples, scarifying the solution quality. For billion scale networks, even one sample will let to an extremely large ILP, that exceeds the capability of the best solvers. Second, the theory developed to assess the solution quality such as those in [11], [12], only provide approximate confidence interval. That is the quality assessment is only justified for sufficiently large samples and may not hold for small sample sizes. And third, most existing solution quality assessment methods [11], [12] only provide the assessment for a given number of sample size. Thus, if the quality requirement is given a priori, e.g., (ϵ, δ) approximation, there is not an efficient algorithmic framework to identify the number of necessary samples.

IV. TipTop: (Almost) Exact Solution in Large-scale Networks

In this section, we introduce our main contribution Tip-Top: a near exact algorithm (*not* approximation algorithm) for CTVM. TipTop is the first exact algorithm that can return a $(1 - \epsilon)$-approximation ratio, overcome the above mentioned scalability issues and can run on billion-scale networks.

A. TipTop Algorithm Overview

For readability, we present our solution to CTVM in which all nodes have uniform cost. Let $BD = k$, we want to find S with $|S| \le k$ so as to maximize the benefit $\mathbb{B}(S)$. The generalization to the more general settings of CTVM is straight forward and omitted due to space limit.

At a high level, TipTop first generates a collection $\mathcal{R}$ of random RR sets which serves as the searching space to find a candidate solution $\hat{S}_k$ to CTVM. It next calls the Verify procedure which independently generates another collection of random RR sets to closely estimate the objective function's value of the candidate solution. If this value is not close enough to the optimal solution, TipTop generates more samples by calling the IncreaseSamples procedure to enlarge the search space, and thus finding another better candidate solution. When the objective function's value of the candidate solution is close enough to the optimal one, TipTop halts and returns the found solution. The pseudo-code of TipTop is presented in Alg. 1.

As shown in Theorem 2, TipTop has an approximation of $(1 - \epsilon)$ with a probability of at least $(1 - \delta)$. The key point to improve the current best ratio of $(1 - 1/e - \epsilon)$ (and $(1 - 1/\sqrt{e} - \epsilon)$) to $(1 - \epsilon)$ lies in solving the Maximum Coverage (MC) problem after generating $\mathcal{R}$ random RR sets of samples. Instead of using greedy technique as in all existing algorithms in the literature, we solve the MC *exactly* using Integer Linear Program (ILP) (detailed in subsection IV-B).

Trade-off: "Better approximation guarantee" vs. "More samples". The adoption of the ILP in the place of the greedy method results in a trade-off between "better approximation guarantee" and "More samples". Given a desired approximation guarantee, say $\rho = 1 - 1/e - \epsilon$, the two factors that influence ρ are 1) the approximation factor by the algorithm

---

Algorithm 1 TipTop Algorithm

Input: Graph $G = (V, E, b, c, w)$, budget $k > 0$, and $\epsilon, \delta \in (0, 1)$.
Output: Seed set S_k.
1: $\Lambda \leftarrow (1 + \epsilon)(2 + \frac{2}{3}\epsilon)\frac{1}{\epsilon^2}\ln\frac{2}{\delta}$
2: $t \leftarrow 1; t_{max} = \lceil 2\ln n/\epsilon\rceil; v_{max} \leftarrow 6$
3: $\Lambda_{max} \leftarrow (1 + \epsilon)(2 + 2/3\epsilon)\frac{2}{\epsilon^2}(\ln\frac{2}{\delta/4} + \ln\binom{n}{k})$
4: Generate random RR sets $R_1, R_2, \ldots$ using BSA [1].
5: **repeat**
6:      $N_t \leftarrow \Lambda \times e^{\epsilon t}; \mathcal{R}_t \leftarrow \{R_1, R_2, \ldots, R_{N_t}\}$
7:      $\hat{S}_k \leftarrow \text{ILP}_{MC}(\mathcal{R}_t, c, k)$
8:      $< passed, \epsilon_1, \epsilon_2 > \leftarrow \text{Verify}(\hat{S}_k, v_{max}, 2^{v_{max}}N_t)$
9:      **if** (not $passed$) and ($Cov_{\mathcal{R}}(\hat{S}_k) \le \Lambda_{max}$)
then $t \leftarrow \text{IncreaseSamples}(t, \epsilon_1, \epsilon_2)$
10: **until** $passed$ or $Cov_{\mathcal{R}}(\hat{S}_k) > \Lambda_{max}$
11: **return** $\hat{S}_k$

---

to solve Max-Coverage problem and 2) the number of samples that decide the quality of approximation for the objective function. On one hand, better approximation guarantee provided by the ILP implies that our approach can obtain the guarantee ρ with less number of samples. On the other hand, the greedy algorithm is a lot faster than the ILP, thus can handle more samples in the same amount of time.

The new adoption of ILP poses new challenges. Solving ILPs is now the bottleneck in terms of both time and memory, dominating those in the verifying counterpart (Verify procedure). Thus it is critical that our approach minimize the number of RRsets used in the ILP to find the candidate solution $\hat{S}_k$. This is different from the previous approaches SSA and D-SSA that attempt to minimize the *total number of RR sets* in both the searching and verifying.

Moreover, both the approaches in SSA and D-SSA have design limitations that make them unsuitable for coupling with the ILP. SSA [6] has fixed parameter settings for searching and verifying and it is difficult to find a good settings for different input ranges. D-SSA has a more flexible approach. However, it is constrained to use the same amount of samples for searching and verifying. Thus, it is also not good for unbalanced searching/verifying.

This enforces us to take a new approach in balancing the samples in searching and verifying. As shown in line 1 of Alg. 1, we start with a smaller set of samples $\mathcal{R}$ of size Λ. Also, we use IncreaseSamples to dynamically adjust the sample size, ensure just enough samples to find a near-optimal candidate solution to CTVM. We will discuss more details about IncreaseSamples later in subsection IV-C.

Secondly, in an effort to keep the ILP size small, we need to avoid executing IncreaseSamples as much as possible and allow IncreaseSamples to add a large batch of samples, if it saw the candidate solution is still far away from the desired. Additionally, we allocate more resource into proving the quality of candidate solutions, which is handled by Verify, described in subsection IV-C.

And finally, as CTVM considers arbitrary benefits, we utilize Benefit Sampling Algorithm – BSA in [1] to embed the benefit of each node into consideration. BSA performs a reversed influence sampling (RIS) in which *the probability of a node chosen as the source is proportional to its benefit*. Random hyperedges generated via BSA can capture the "benefit landscape". That is they can be used to estimate the benefit of any seed set S as stated in the following lemma.

Lemma 2. *[1] Given a fixed seed set $S \subseteq V$, and let $R_1, R_2, \ldots, R_j, \ldots$ be random RR sets generated using Benefit Sampling Algorithm [1], define random variables*

$$Z_j = \begin{cases} 1 & \text{if } R_j \cap S \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

then

$$\mathbb{E}[Z_j] = \Pr[R_j \cap S \neq \emptyset] = \frac{\mathbb{B}(S)}{\Gamma} \tag{15}$$

where $\Gamma = \sum_{v \in V} b(v)$ is the total nodes' benefit.

For a collection of T random RR sets $\mathcal{R} = \{R_1, R_2, \ldots, R_T\}$, we denote $Cov_{\mathcal{R}}(S) = \sum_{j=1}^{T} Z_j$ as the number of RR sets that intersect S, and $\mathbb{B}_{\mathcal{R}}(S) = \frac{Cov_{\mathcal{R}}(S)}{T} \times \Gamma$ as the estimation of $\mathbb{B}(S)$ via $\mathcal{R}$. We discuss the rest of TIPTOP in the following subsections.

B. Integer Linear Programming ILP_{MC}

Given a collection of RR sets $\mathcal{R}$, we formulate the following $ILP_{MC}(\mathcal{R}, c, BD)$, to find the optimal solution over the generated RR sets to the MC problem, assuming uniform cost for node selection.

$$\text{maximize} \sum_{R_j \in \mathcal{R}} (1 - y_j) \tag{16}$$

$$\text{subject to} \sum_{v \in V} s_v \leq k \tag{17}$$

$$\sum_{v \in R_j} s_v + y_j \geq 1 \qquad \forall R_j \in \mathcal{R} \tag{18}$$

$$y_j, s_i \in \{0, 1\} \tag{19}$$

Here, $s_v = 1$ iff node v is selected into the seed set, and $s_v = 0$, otherwise. The variable $y_j = 1$ indicates that the RR sets R_j cannot be covered by the seed set ($S = \{v | s_v = 1\}$) and $y_j = 0$, otherwise. The objective aims to cover as many RR sets as possible while keeping the cost at most k using the constraint (17).

We note that the benefit in selecting the node $b(u)$ does not appear in the above ILP as it is embedded in the Benefit Sampling Algorithm (BSA) in [1].

On one hand, the above ILP can be seen as an SAA of the CTVM problem as it attempts to find optimal solutions over the randomly generated samples. On the other hand, it is different from the traditional SAA discussed in Sec. III as it does not require the realization of all random variables, i.e., the status of the edges. Instead, only a local portion of the graph surrounding the sources of RR sets need to be revealed. This critical difference from traditional SAA significantly reduces the size of each sample, effectively, resulting in a much more compact ILP.

C. The Verify and IncreaseSamples Procedures

As shown in Alg. 2, Verify takes a candidate solution $\hat{S}_k$, precision limit v_{max}, and the maximum number of RR sets T_{cap} as the input. It keeps generate RR sets to estimate $\mathbb{B}(\hat{S}_k)$ until either the relative error reaches $\epsilon/2^{v_{max}-1}$ or the maximum number of generated samples T_{cap} is reached.

Verify uses the stopping rule algorithm in [6] to estimate the influence. It generates a new pool of random RR sets, denoted by $\mathcal{R}_{ver}$. For each pair ϵ_2', δ_2', derived from ϵ_2 and δ_2, the stopping rule algorithm will stop when either the cap T_{max} is reached or there is enough evidence (i.e. $cov \geq \Lambda_2$) to conclude that

$$Pr[(1 - \epsilon_2')\mathbb{B}(\hat{S}_k) \leq \mathbb{B}_{\mathcal{R}_{ver}}(\hat{S}_k) \leq (1 + \epsilon_2')\mathbb{B}(\hat{S}_k)] \geq 1 - \delta_2'$$

The value of δ_2' is selected as in line 3 so that the probability of the union of all the bad events is bounded by δ_2.

If the stopping rule algorithm stops within T_{cap} RR sets, the algorithm evaluates the relative difference ϵ_1 between the estimations of $\mathbb{B}(\hat{S}_k)$ via $\mathcal{R}_t$ and $\mathcal{R}_{ver}$. It also estimates the relative gap ϵ_3 between $\mathbb{B}(S_k^*)$ and its estimation using $\mathcal{R}_t$. If the combined gap $(1 - \epsilon_1)(1 - \epsilon_2)(1 - \epsilon_3) < (1 - \epsilon)$, Verify returns 'true' and goes back to TIPTOP. In turn, TIPTOP will return $\hat{S}_k$ as the solution and terminate.

If $\hat{S}_k$ does not pass the check in Verify, TIPTOP uses the sub-procedure IncreaseSamples (Alg. 3) to increase the size of the RR sets. Having more samples will likely lead to better candidate solution $\hat{S}_k$, however, also increase the IP solving time. Instead of doubling the current set $\mathcal{R}$ as SSA does, we carefully use the information in the values of ϵ_1 and ϵ_2 from the previous round to determine the increase in the sample sizes. Recall that the sample size is $e^{t\epsilon}\Lambda$ for increasing integer t. Thus, we increase the sample size via increasing t by (approximately) $\log_{e^\epsilon} \frac{\epsilon_1^2}{\epsilon^2}$. We force t to increase by at least one and at most $\Delta t_{max} = \lceil 2/\epsilon \rceil$. That is the number of samples will increase by a multiplicative factor between $e^\epsilon \approx (1 + \epsilon)$ and $e^{\Delta t_{max}} \approx e^2$.

Algorithm 2 Verify

Input: Candidate solution $\hat{S}_k$, v_{max}, and T_{cap}.
Output: Passed/not passed, ϵ_1, and ϵ_2.
1: $\quad \mathcal{R}_{ver} \leftarrow \emptyset, \delta_2 = \frac{\delta}{4}, cov = 0, \epsilon_1 = \epsilon_2 = \infty$
2: $\quad$ **for** $i \leftarrow 0$ to $v_{max} - 1$ **do**
3: $\qquad \epsilon_2 = \min\{\epsilon, 1\}/2^i, \epsilon_2' = \frac{\epsilon_2}{1 - \epsilon_2}; \delta_2' = \delta_2/(v_{max} \times t_{max})$
4: $\qquad \Lambda_2 = 1 + (2 + 2/3\epsilon_2')(1 + \epsilon_2') \ln \frac{2}{\delta_2'} \frac{1}{(\epsilon_2')^2}$
5: $\qquad$ **while** $cov < \Lambda_2$ **do**
6: $\qquad\quad$ Generate R_j with BSA [1] and add it to $\mathcal{R}_{ver}$
8: $\qquad\quad$ **if** $R_j \cap S \neq \emptyset$ **then** $cov = cov + 1$
9: $\qquad\quad$ **if** $|\mathcal{R}_{ver}| > T_{cap}$ **then** return $< false, \epsilon_1, 2\epsilon_2 >$
10: $\qquad$ **end while**
11: $\qquad \mathbb{B}_{ver}(\hat{S}_k) \leftarrow \Gamma \frac{cov}{|\mathcal{R}_{ver}|}, \epsilon_1 \leftarrow 1 - \frac{\mathbb{B}_{ver}(\hat{S}_k)}{\mathbb{B}_{\mathcal{R}}(\hat{S}_k)}$
12: $\qquad$ **if** $(\epsilon_1 > \epsilon)$ **then** return $< false, \epsilon_1, \epsilon_2 >$
13: $\qquad \epsilon_3 \leftarrow \sqrt{\frac{3 \ln(t_{max}/\delta_1)}{(1 - \epsilon_1)(1 - \epsilon_2)Cov_{\mathcal{R}_t}(\hat{S}_k)}}.$
14: $\qquad$ **if** $(1 - \epsilon_1)(1 - \epsilon_2)(1 - \epsilon_3) > (1 - \epsilon)$ **then**
$\qquad\qquad$ return $< true, \epsilon_1, \epsilon_2 >$
15: $\quad$ **end for**
16: return $< false, \epsilon_1, \epsilon_2 >$

Algorithm 3 IncreaseSamples

Input: $t, \epsilon_1, \epsilon_2$.
Action: Return iteration t.
1: $\quad \Delta t_{max} = \lceil 2/\epsilon \rceil$
2: $\quad$ return $t + \min\{\max\{\lceil 1/\epsilon \ln \frac{\epsilon_1^2}{\epsilon^2} \rceil, 1\}, \Delta t_{max}\}$

V. OPTIMALITY OF TIPTOP

In this section, we prove that TIPTOP returns a solution $\hat{S}$ that is optimal up to a multiplicative error $1 - \epsilon$ with high probability. Fig. 1 shows the proof map of our main Theorem.

Let $R_1, R_2, R_3, \ldots, R_j, \ldots$ be the random RR sets generated in TIPTOP. Given a seed set S, define random variables Z_j as in (14) and $Y_j = Z_j - \mathbb{E}[Z_j]$. Then Y_j satisfies the conditions of a *martingale* [19], i.e., $\mathbb{E}[Y_i | Y_1, Y_2, \ldots, Y_{i-1}] = Y_{i-1}$ and $\mathbb{E}[Y_i] < +\infty$. This martingale view is adopted from [5] to cope with the fact that random RR sets might not be independent due to the stopping condition: the later RR sets

are generated only when the previous ones do not satisfy the stopping conditions. We obtain the same results in Corollaries 1 and 2 in [5].

Lemma 3 ([5]). *Given a set of nodes S and random RR sets $\mathcal{R} = \{R_j\}$ generated in TIPTOP, define random variables Z_j as in (14). Let $\mu_Z = \frac{\mathbb{B}(S)}{\Gamma}$ and $\hat{\mu}_Z = \frac{1}{T}\sum_{i=1}^{T} Z_i$ be an estimation of μ_Z, for fixed $T > 0$. For any $0 \leq \epsilon$, the following inequalities hold*

$$\Pr[\hat{\mu} \geq (1+\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{2+\frac{2}{3}\epsilon}}, \text{ and} \quad (20)$$

$$\Pr[\hat{\mu} \geq (1+\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{3}}, \text{ and} \quad (21)$$

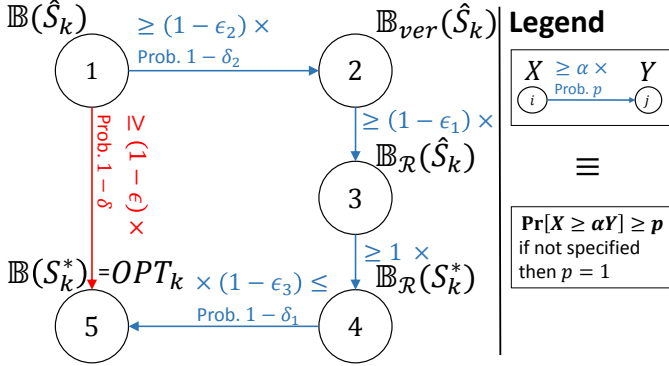$$\Pr[\hat{\mu} \leq (1-\epsilon)\mu] \leq e^{\frac{-T\mu\epsilon^2}{2}}. \quad (22)$$

Fig. 1: Proof map of the main Theorem 2.

A common framework in [5], [6], [20] is to generate random RR sets and use the greedy algorithm to select k seed nodes that cover most of the generated RR sets. It is shown in Lemma 3 [20] that $(1 - 1/e - \epsilon)$ approximation algorithm with probability $1 - \delta$ is obtained when the number of RR sets reach a threshold

$$\theta(\epsilon, \delta) = c \times (8 + \epsilon) \left(\ln \binom{n}{k} + \ln \frac{2}{\delta}\right) \frac{n}{OPT_k} \frac{1}{\epsilon^2}, \quad (23)$$

for some constant $c > 0$.

The constant c is bounded to be $8+\epsilon$ in [20], brought down to $8(e-2)(1-1/(2e))^2 \approx 3.7$ in [1] . And the current best is $c = 2 + 2/3\epsilon$, inducted from Lemma 6 in [5].

Note that the threshold $\theta(\epsilon, \delta)$ cannot be used to decide how many RR sets we need to generate. The reason is that θ depends on the unknown value OPT_k, of which computation is #P-hard.

To overcome that hurdle, a simple stopping rule is developed in [1] to check whether we have sufficient RR sets to guarantee, w.h.p., a $(1 - 1/e - \epsilon)$ approximation. The rule is that we can stop when we can find *any* seed set S_k, of size k, that coverage $Cov_{\mathcal{R}}(S_k)$ exceeds

$$\Lambda_{max} = (1+\epsilon)\theta(\epsilon, \delta/4) \times \frac{OPT_k}{n} \quad (24)$$

$$= (1+\epsilon)(2+2/3\epsilon)\frac{1}{\epsilon^2}(\ln\frac{2}{\delta/4} + \ln\binom{n}{k}) \quad (25)$$

Our algorithm TIPTOP utilizes this stopping condition to guarantee that at most $\Omega(\theta(\epsilon, \delta))$ RR sets are used in the ILP in the worst-case. As a result, the size of the ILP is kept to be almost linear size, assuming $k \ll n$

Theorem 1. *The expected number of non-zeros in the ILP of*

TIPTOP *is*

$$\Omega\left(\left(\ln\binom{n}{k} + \ln\frac{2}{\delta}\right)\frac{n}{\epsilon^2}\right)$$

Proof: The expected number of RR sets is $\Omega((1 + \epsilon)\theta(\epsilon, \delta))$. Moreover, the expected size of each RR sets is upper-bounded by $\frac{OPT_k}{k}$ (Lemma 4 and Eq. (7) in [20]). Thus, the size of the ILP which is equal the total sizes of all the RR sets plus n, the size of the cardinality constraint, is at most $\Omega((1+\epsilon)\theta(\epsilon, \delta)OPT_k) = \Omega\left(\left(\ln\binom{n}{k} + \ln\frac{2}{\delta}\right)\frac{n}{\epsilon^2}\right)$ ∎

In practice, the ILP size is much smaller than the worst-case bound in the above theorem, as shown in Section VI.

Lemma 4. *Let S_k^* be a seed set of size k with maximum benefit, i.e., $\mathbb{B}(S_k^*) = OPT_k$. Denote by $\mu_k^* = \frac{OPT_k}{\Gamma}, \delta_1 = \delta/4$ and $\epsilon_t^* = \sqrt{\frac{3\ln(t_{max}/\delta_1)}{N_t\mu_t^*}}$. We have:*

$$\Pr[\mathbb{B}_{\mathcal{R}_t}(S_k^*) \geq (1 - \epsilon_t^*)\mathbb{B}(S_k^*) \; \forall \mathbf{t} = \mathbf{1}..\mathbf{t_{max}}] \geq 1 - \delta_1$$

Proof: Apply Lem. 3 (Eq. 21) for seed set S_k^*, mean μ_k^*, ϵ_t^* and N_t samples. For each $t \in [1..t_{max}]$, after some algebra we obtain:
$$\Pr[\mathbb{B}_{\mathcal{R}_t}(S_k^*) < (1 - \epsilon_t)\mathbb{B}(S_k^*)] < \delta_1/t_{max}.$$

Taking the union bound over all $t \in [1, t_{max}]$ yields the proof. ∎

Lemma 5. *[$(1)\to(2)$] The probability of the bad event that there exists some set of $t \in [1, t_{max}], i \in [0, v_{max} - 1]$, and $\epsilon_2 = \epsilon/2^i$ so that Verify returns some bad estimation of $\hat{S}_k$ at line 10, i.e.,*

$$\mathbb{B}_{\mathcal{R}_{ver}}(\hat{S}_k) > \frac{1}{(1-\epsilon_2)}\mathbb{B}(\hat{S}_k),$$

is less than δ_2. Here $\mathbb{B}_{ver}(\hat{S}_k) = \Gamma \times \frac{Cov_{\mathcal{R}_{ver}}(\hat{S}_k)}{|\mathcal{R}_{ver}|}$ be an estimation of $\mathbb{B}(\hat{S}_k)$ using random RR sets in $\mathcal{R}_{ver}$.

Proof: After reaching line 10 in Verify, the generated RR sets within Verify, denoted by $\mathcal{R}_{ver}$, will satisfy the condition that
$$cov = Cov_{\mathcal{R}_{ver}}(\hat{S}_k) \geq \Lambda_2 = 1 + (2 + \frac{2}{3}\epsilon_2')(1 + \epsilon_2')\ln\frac{2}{\delta_2'}\frac{1}{\epsilon_2'^2},$$
where $\epsilon_2' = \frac{\epsilon_2}{1-\epsilon_2}$.

According to the stopping rule theorem[2] in [21], this stopping condition guarantees that
$$\Pr[\mathbb{B}_{\mathcal{R}_{ver}}(\hat{S}_k) > (1 + \epsilon_2')\mathbb{B}(\hat{S}_k)] < \delta_2'$$

Substitute $\epsilon_2' = \frac{\epsilon_2}{1-\epsilon_2}$ and simplify, we obtain
$$\Pr[\mathbb{B}_{\mathcal{R}_{ver}}(\hat{S}_k) > \frac{1}{1-\epsilon_2}\mathbb{B}(\hat{S}_k)] < \delta_2'$$

The number of times Verify invoked the stopping condition to estimate $\hat{S}_k$ is at most $t_{max} \times v_{max}$. Thus, we can use the union bound of all possible bad events to get a lower-bound $\delta_2' \times t_{max} \times v_{max} = \delta_2$ for the probability of existing a bad estimation of $\hat{S}_k$. ∎

$(2)\to(3)$: This holds due to the definition of ϵ_1 in Verify. Since $\epsilon_1 \leftarrow 1 - \mathbb{B}_{ver}(\hat{S}_k)/\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k)$, it follows that
$$\mathbb{B}_{ver}(\hat{S}_k) = (1 - \epsilon_1)\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k).$$

We note that it is possible that $\epsilon_1 < 0$, and the whole proof still goes through even for that case. In the experiments, we, however, do not observe negative values of ϵ_1.

---

[2]replacing the constant $4(e-2)$ with $2+2/3\epsilon$ due to the better Chernoff-bound in Lemma 3

$(3)\to(4)$: Since $\hat{S}_k$ is an optimal solution of $\text{ILP}_{MC}(\mathcal{R}, c, B)$, it will be the k-size seed set that intersects with the maximum number of RR sets in $\mathcal{R}$. Let S_k^* be an optimal k-size seed set, i.e., the one that results in the maximum expected benefit[3]. Since $|S_k^*| = k$, it follows that

$$\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k) \geq \mathbb{B}_{\mathcal{R}_t}(S_k^*),$$

where $\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k)$ and $\mathbb{B}_{\mathcal{R}_t}(S_k^*)$ denote the number of RR sets in $\mathcal{R}_t$ that intersect with $\hat{S}_k$ and S_k^*, respectively.

Theorem 2. *[Main theorem $(1)\to(5)$] Let $\hat{S}_k$ be the solution returned by* TIPTOP *(Algorithm 1). We have:*

$$\Pr[\mathbb{B}(\hat{S}_k) \geq (1-\epsilon)OPT_k] \geq 1 - \delta \quad (26)$$

Proof: First we use the union bound to bound the probability of the bad events. Then we show if none of the bad events happen, the algorithm will return a solution satisfying $\mathbb{B}(\hat{S}_k) \geq (1-\epsilon)OPT_k$.

The bad events and the bounds on their probabilities are
1) $\Pr[\exists t : \mathbb{B}_{\mathcal{R}_t}(S_k^*) < (1-\epsilon_t^*)\mathbb{B}(S_k^*)] < \delta_1$ (Lem. 4)
2) $\Pr[\exists t, i, \epsilon_2 = \frac{\epsilon}{2^i} : \mathbb{B}_{\mathcal{R}_{ver}}(\hat{S}_k) > \frac{1}{1-\epsilon_2}\mathbb{B}(\hat{S}_k)] < \delta_2$ (Lem. 5)
3) $\Pr[(Cov_{\mathcal{R}_t}(\hat{S}_k) > \Lambda_{max})$ and $(\mathbb{B}(\hat{S}_k) < (1-1/e-\epsilon)OPT_k)] < \delta/4$
4) $\Pr[(t > t_{max})$ and $(Cov_{\mathcal{R}_t}(\hat{S}_k) \leq \Lambda_{max})] < \delta/4$

The bounds in 1) and 2) come directly from Lems. 4 and 5. The bound in 3) is a direct consequence of the stopping condition algorithm in [1]. The bound in 4) can be shown by noticing that when $t > t_{max}$ then $N_t \gg \theta(\epsilon, \delta)$.

Apply the union bound; the probability that none of the above bad events happens is at most $\delta_1 + \delta_2 + \delta/4 + \delta/4 = \delta$.

Assume that none of the above bad events happen, we shall show that the algorithm returns a $(1-\epsilon)$ optimal solution. If TIPTOP stops with $Cov_{\mathcal{R}_t}(\hat{S}_k) > \Lambda_{max}$, it is obvious that $\mathbb{B}(\hat{S}_k) \geq (1-\epsilon)OPT_k$, since the bad event in 3) do not happen. Otherwise, algorithm Verify returns 'true' at line 14 for some $t \in [1, t_{max}]$ and $i \in [0, v_{max})$. Follow the path $(1)\to(2)\to(3)\to(4)$. No bad event in 2) implies

$$\mathbb{B}(\hat{S}_k) \geq (1-\epsilon_2)\mathbb{B}_{ver}(\hat{S}_k) \quad (27)$$
$$\geq (1-\epsilon_2)(1-\epsilon_1)\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k) \quad (28)$$
$$\geq (1-\epsilon_2)(1-\epsilon_1)\mathbb{B}_{\mathcal{R}_t}(S_k^*) \quad (29)$$

No bad event in 1) implies
$$\mathbb{B}_{\mathcal{R}_t}(S_k^*) \geq (1-\epsilon_t^*)\mathbb{B}(S_k^*)$$

$(4)\to(5)$: To show
$$\mathbb{B}_{\mathcal{R}_t}(S_k^*) \geq (1-\epsilon_3)\mathbb{B}(S_k^*),$$

We prove that
$$\epsilon_t^* = \sqrt{\frac{3\ln(t_{max}/\delta_1)}{N_t \mu_t^*}}$$
$$\leq \epsilon_3 = \sqrt{\frac{3\ln(t_{max}/\delta_1)}{(1-\epsilon_1)(1-\epsilon_2)Cov_{\mathcal{R}_t}(\hat{S}_k)}}$$
$$\Leftrightarrow N_t \mu_t^* \geq (1-\epsilon_1)(1-\epsilon_2)Cov_{\mathcal{R}_t}(\hat{S}_k)$$
$$\Leftrightarrow N_t \mathbb{B}(S_k^*)/\Gamma \geq (1-\epsilon_1)(1-\epsilon_2)N_t \mathbb{B}_{\mathcal{R}_t}(\hat{S}_k)/\Gamma$$
$$\Leftrightarrow \mathbb{B}(S_k^*) \geq (1-\epsilon_1)(1-\epsilon_2)\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k)$$

---

[3]If there are multiple optimal solutions, we break the tie by using alphabetical order on nodes' ids.

The last one holds due to the optimality of S_k^* and Eq. 28.

$$\mathbb{B}(S_k^*) \geq \mathbb{B}(\hat{S}_k) \geq (1-\epsilon_1)(1-\epsilon_2)\mathbb{B}_{\mathcal{R}_t}(\hat{S}_k)$$

Combine Eq. 29 and $(4)\to(5)$, we have

$$\mathbb{B}(\hat{S}_k) \geq (1-\epsilon_2)(1-\epsilon_1)\mathbb{B}_{\mathcal{R}_t}(S_k^*)$$
$$\geq (1-\epsilon_2)(1-\epsilon_1)(1-\epsilon_3)\mathbb{B}(S_k^*) \geq (1-\epsilon)OPT_k$$

The last one holds due to the terminating condition $(1-\epsilon_2)(1-\epsilon_1)(1-\epsilon_3) > (1-\epsilon)$ in line 14 in Verify. ∎

VI. EXPERIMENTS

We conduct several experiments to illustrate the performance and utility of TIPTOP. First, the performance of TIPTOP is compared to the T-EXACT ILP. Our results show that it is magnitudes faster while maintaining solution quality (sec. VI-A). We then apply TIPTOP as a benchmark for existing methods, showing that they perform better than their theoretical guarantee in certain cases but have significantly degraded performance in others. Finally, we conclude the section with an in-depth analysis of TIPTOP's performance characteristics.

We implemented TIPTOP in C++ using CPLEX to solve the IP. Unless noted otherwise, each experiment is run 10 times and the results averaged. Settings for ϵ are listed with each experiment, but δ is fixed at $1/n$. All influence values are obtained by running a separate estimation program with $\epsilon = 0.02$ using the seed sets produced by each algorithm as input.

Dataset	Network Type	Nodes	Edges
US Pol. Books [22]	Recommendation	105	442
GR-QC [23]	Collaboration	5242	14496
Wiki-Vote [23]	Voting	7115	103689
NetPHY [24]	Collaboration	37149	180826
Twitter [25]	Social	41M	1.5B

TABLE I: Networks used in our experiments.

A. Comparison to the T-EXACT IP

Since T-EXACT produces exact results for IM, we use it show the optimality of TIPTOP. As T-EXACT suffers from scalability issues, we run on the 105-node US Pol. Books network only. As shown in Fig. 2, TIPTOP consistently performs as well as T-EXACT while performing more than 100 times faster. We exploit this behavior to place an upper bound on the performance of approximation methods in the next section.
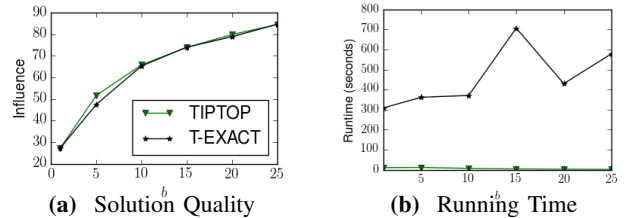

(a) Solution Quality   **(b)** Running Time

Fig. 2: The performance of TIPTOP ($\epsilon = 0.02, \delta = 1/n$) and T-EXACT ($T = 5000$) on the US Political Books [22] network under the cost-aware setting with normalized costs as the budget b is varied.

B. Benchmarking Greedy Methods

As shown above, TIPTOP is capable of producing almost exact solutions significantly faster than other IP-based solutions. This property allows us to bound the performance of approximation methods, which in turn allows us to compare their *absolute* performance rather than merely their *relative* performance. We evaluate the performance of IMM, BCT, and SSA under three problem settings on three networks. All evaluations are under the IC model with edge weights set to $w(u, v) = 1/d_{in}(v)$, where $d_{in}(v)$ is the in-degree of v.

The first setting we consider is the traditional IM problem (which we term *Unweighted* to differentiate it from subsequent settings). We directly apply each author's implementation to this problem with no modifications. Fig. 3a shows that the decade-or-so of work on this problem has resulted in greedy solutions that are much better than their approximation ratio of $(1-1/e-\epsilon)$. Somewhat surprisingly, we find similar results on the *Cost-Aware* problem setting (fig. 3b).

The Cost-Aware setting generalizes the Unweighted setting by adding a cost to each node on the network. In a social network setting, the costs can be understood as the relative amount of payment each user requires to become a part of an advertising campaign (i.e. a seed node). Note that neither IMM nor SSA support costs natively. We extend both to this problem by scaling their objective functions by cost and limiting the number of selected nodes by the sum of costs instead of the number of nodes, which gives each an approximation guarantee of $1 - 1/\sqrt{e} - \epsilon$ [16]. We consider three ways users may decide on their relative value. First, users may define the value of their influence independently from network topology. We model this scenario by assigning costs according to a uniform random distribution on $[0, 1)$.
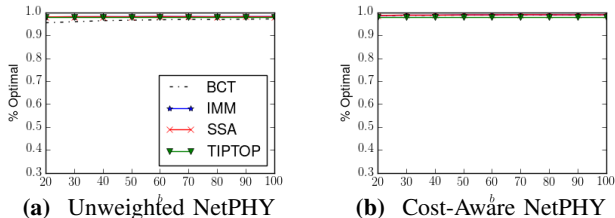

(a) Unweighted NetPHY   **(b)** Cost-Aware NetPHY

Fig. 3: Mean performance of each approximation algorithm as the budget is varied under the Unweighted and Cost-Aware problem settings with $\epsilon = 0.02$. OPT is estimated assuming that TIPTOP achieves exactly $(1 - \epsilon)OPT$.

Alternately, users may use the most readily-available metric to determine their influence: follower count. We consider both linear- and log-scaled functions of this metric as rough upper and lower bounds. In the linear case ($cost(u) = n/|E|d_{out}(u)$; $d_{out}(u)$ is the out-degree of u, and the $n/|E|$ term normalizes the cost), if a user with 2K followers requires \$800 to be influenced, then a user with 2M followers would require \$800K. However, due to cascades increasing the effective influence of users with follower counts between the two, the high-follower users may opt to reduce their prices. The log-scaled function models an extreme case, with the two-million-follower user requiring a mere \$1,527. We show that the performance of all methods remains similar when scaling is changed, though the performance of IMM and SSA drops significantly when costs are randomized.

Our final setting is the CTVM problem described above. We target 5% of each network at random, assigning each targeted user a benefit on $[0.1, 1)$ and each non-targeted user

a benefit of 0. Fig. 4 shows the performance on the GR-QC, Wiki-Vote, and NetPHY networks. Note that neither IMM nor SSA can be extended to the CTVM problem without significant modifications, and therefore each ignores the benefit values.

From Fig. 4, we can see that the topology has a significant impact on the performance of each algorithm. Note that on GR-QC and NetPHY, the performance of BCT relative to the optimal tends downward while on Wiki-Vote it tends upward. Further, on NetPHY both IMM and SSA do astonishingly well despite being ignorant of the targeted nodes. Figure 5 shows that BCT performs similarly regardless of cost function, while the modified IMM and SSA face a significant drop in performance when the costs are divorced from topology.

Coverage	Unweighted	Cost-Aware	CTVM
IMM	6.94×10^6	3.96×10^6	8.59×10^6
SSA	2.01×10^6	2.81×10^6	3.42×10^6
BCT	9.34×10^6	4.15×10^6	6.23×10^6
TIPTOP	2.08×10^3	1.30×10^4	6.04×10^3
Verification	Unweighted	Cost-Aware	CTVM
SSA	4.02×10^7	8.44×10^7	4.02×10^7
TIPTOP	6.25×10^6	1.86×10^9	2.80×10^7

TABLE II: Mean required samples for each algorithm. Approximation guarantees are 0.61 for greedy methods in the unweighted case and 0.37 for the cost-aware case (and the CTVM case for BCT). The guarantee for TIPTOP is 0.6 in all cases. *Coverage*: Samples input into the MC solver. *Verification*: Samples used to verify solution quality. IMM and BCT do not incorporate a verification stage.

C. Analyzing TIPTOP's performance

We now turn our attention to the details of TIPTOP's performance. Fig. 6 compares the running time of each method as the approximation guarantee and budget are varied. From this, we can see that the runtime performance of TIPTOP is not far from – and in some cases better than – greedy methods. With a comparable guarantee, TIPTOP is as fast as SSA and an order of magnitude faster than IMM. However, as the guarantee tends to 1, the advantage is lost. Fig. 6b shows that while the runtime of both IMM and BCT is smoothly linear and for SSA is near-constant, the running time of TIPTOP is less predictable. This is the cost of solving an IP instead of greedily solving MC: the IP solver may quickly find the optimal solution, or it may take exponentially long to do so. However, on average the running time of TIPTOP is low enough that it is practical to run even on very large networks. To further establish the scalability of TIPTOP, we run it on the Twitter dataset (fig. 7). While the running time of TIPTOP is worse, it still produces 98% optimal solutions *in under three hours*. Previously, solving an IP on a network with millions of nodes and billions of edges had been thought impractical. TIPTOP is able to accomplish this by dramatically reducing the number of samples used to solve MC. Table II shows that on average, TIPTOP uses on the order of 10^3 fewer samples than greedy methods to solve MC. As SSA has a Stare phase in which it verifies the quality of a generating solution, we also compare the number of samples used in our method and SSA. The lower section of table II shows that even then, the number of samples used to verify the approximation guarantee is similar. The total number of samples used in both phases of TIPTOP, in general, in smaller that that of SSA. Further, the approximation ratio of TIPTOP affords greater precision if one
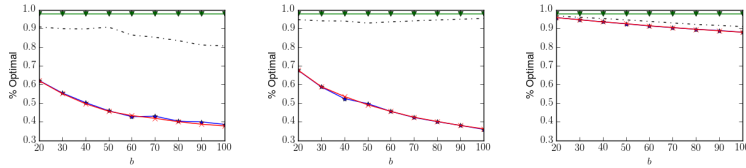
**(a)** GR-QC   **(b)** Wiki-Vote   **(c)** NetPHY

Fig. 4: Mean performance of each approximation algorithm as the budget is varied under the CTVM problem setting with linear costs.
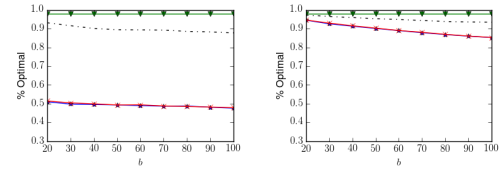
**(a)** Uniform Random   **(b)** Log-Outdegree

Fig. 5: Performance on the NetPHY network under the CTVM setting with alternate costs.

needs to guarantee a certain level of performance, as shown in Fig. 7. Where prior work produced results of similar quality regardless of ϵ, TIPTOP enables setting guarantees to exactly the required value. If, for instance, an application requires an 80% approximation ratio, ϵ can be set to exactly 0.2, at which point TIPTOP is as fast as state-of-the-art greedy solutions.
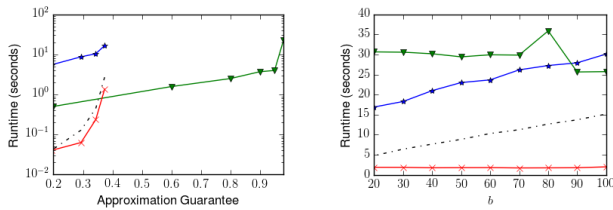
(a) Runtime as ϵ is increased. b is held constant at 50.   **(b)** Runtime as b is increased. ϵ is held constant at 0.02.

Fig. 6: Mean running time of TIPTOP, IMM, SSA, and BCT on the NetPHY network under the CTVM problem setting.
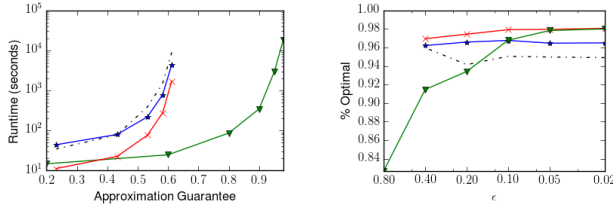
(a) Running Time   **(b)** Solution Quality

Fig. 7: Mean results on the Twitter network under the Unweighted setting. Only one repetition is used on this dataset.

VII. CONCLUSION

In this paper, we propose the first (almost) exact solutions to the CTVM (and thus IM) problem, namely T-EXACT and TIPTOP. T-EXACT uses the traditional stochastic programming approach and thus suffers the scalability issue. In contrast, our TIPTOP with innovative techniques in reducing the number of samples to meet the requirement of ILP solver is able to run on billion-scale OSNs such as Twitter under three hours. As TIPTOP is an exact solution to the ratio of $(1 - \epsilon)$, it significantly improves from the current best ratio $(1 - 1/e - \epsilon)$ for IM and $(1 - 1/\sqrt{e} - \epsilon)$ for CTVM. TIPTOP also lends a tool to benchmark the absolute performance of existing algorithms on large-scale networks.

REFERENCES

[1] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "Cost-aware targeted viral marketing in billion-scale networks," in *INFOCOM*. IEEE, 2016.

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

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

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

[5] Y. Tang, Y. Shi, and X. Xiao, "Influence Maximization in Near-Linear Time: A Martingale Approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. ACM, pp. 1539–1554.

[6] H. T. Nguyen, M. T. Thai, and T. N. Dinh, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks," in *SIGMOD*. ACM, 2016.

[7] T. N. Dinh, H. Zhang, D. T. Nguyen, and M. T. Thai, "Cost-effective viral marketing for time-critical campaigns in large-scale social networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 6, pp. 2001–2011, 2014.

[8] H. Zhang, T. N. Dinh, and M. T. Thai, "Maximizing the spread of positive influence in online social networks," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. IEEE, 2013, pp. 317–326.

[9] T. N. Dinh, D. T. Nguyen, and M. T. Thai, "Cheap, easy, and massively effective viral marketing in social networks: truth or fiction?" in *Proceedings of the 23rd ACM conference on Hypertext and social media*. ACM, 2012, pp. 165–174.

[10] H. Zhang, D. T. Nguyen, H. Zhang, and M. T. Thai, "Least cost influence maximization across multiple social networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 929–939, 2016.

[11] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modeling and Theory, Second Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2014.

[12] G. Bayraksan and P. D. Morton, "Assessing solution quality in stochastic programs," *Mathematical Programming*, vol. 108, no. 2, pp. 495–514, 2006.

[13] U. Feige, "A threshold of ln n for approximating set cover," *Journal of ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[14] N. P. Nguyen, Y. Xuan, and M. T. Thai, "A novel method for worm containment on dynamic social networks," in *Military Communications Conference*, 2010, pp. 2180–2185.

[15] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '14. SIAM, 2014, pp. 946–957.

[16] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information Processing Letters*, vol. 70, no. 1, pp. 39–45, 1999.

[17] H. Nguyen and R. Zheng, "On budgeted influence maximization in social networks," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 6, pp. 1084–1094, 2013.

[18] A. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[19] F. Chung and L. Lu, "Concentration inequalities and martingale inequalities: a survey," *Internet Mathematics*, vol. 3, no. 1, pp. 79–127, 2006.

[20] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 75–86.

[21] P. Dagum, R. Karp, M. Luby, and S. Ross, "An optimal algorithm for monte carlo estimation," *SIAM J. Comput.*, vol. 29, no. 5, pp. 1484–1496, Mar. 2000.

[22] V. Krebs, "Books about us politics," unpublished, compiled by M. Newman. Retrieved from http://www-personal.umich.edu/~mejn/netdata/.

[23] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[24] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD '09*.   New York, NY, USA: ACM, 2009, pp. 199–208.

[25] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*.   ACM, 2010, pp. 591–600.