Unconstrained Submodular Maximization with Modular Costs: Tight Approximation and Application to Profit Maximization

Tianyuan Jin
National University of Singapore
tianyuan1044@gmail.com

Yu Yang
City University of Hong Kong
yuyang@cityu.edu.hk

Renchi Yang
National University of Singapore
renchi@nus.edu.sg

Jieming Shi*
Hong Kong Polytechnic University
jieming.shi@polyu.edu.hk

Keke Huang
National University of Singapore
kkhuang@nus.edu.sg

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

ABSTRACT

Given a set V, the problem of *unconstrained submodular maximization with modular costs (USM-MC)* asks for a subset $S \subseteq V$ that maximizes $f(S) - c(S)$, where f is a non-negative, monotone, and submodular function that gauges the utility of S, and c is a non-negative and modular function that measures the cost of S. This problem finds applications in numerous practical scenarios, such as profit maximization in viral marketing on social media.

This paper presents ROI-Greedy, a polynomial time algorithm for USM-MC that returns a solution S satisfying $f(S) - c(S) \geq f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)$, where S^* is the optimal solution to USM-MC. To our knowledge, ROI-Greedy is the first algorithm that provides such a strong approximation guarantee. In addition, we show that this worst-case guarantee is *tight*, in the sense that no polynomial time algorithm can ensure $f(S) - c(S) \geq (1+\epsilon) \cdot \left(f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)\right)$, for any $\epsilon > 0$. Further, we devise a non-trivial extension of ROI-Greedy to solve the profit maximization problem, where the precise value of $f(S)$ for any set S is unknown and can only be approximated via sampling. Extensive experiments on benchmark datasets demonstrate that ROI-Greedy significantly outperforms competing methods in terms of the trade-off between efficiency and solution quality.

PVLDB Reference Format:
Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Unconstrained Submodular Maximization with Modular Costs: Tight Approximation and Application to Profit Maximization. PVLDB, 14(10): 1756-1768, 2021.
doi:10.14778/3467861.3467866

1 INTRODUCTION

Let V be a set and 2^V be the power set of V. A function $f : 2^V \rightarrow \mathbb{R}$ is *submodular*, if for any $A \subseteq B \subseteq V$ and any $u \in V$,

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B),$$

i.e., when f's input is sizable, adding an element u to the input of f tends to have a lesser effect on the value of f. Submodular functions are commonly used to model problems with *diminishing returns*, and have been a subject of active research for the past few decades.

In this paper, we study the optimization of submodular functions in an intuitive cost-based setting as follows:

- f is non-negative, monotone, and submodular, and
- each element e in V has a positive *cost*, and
- we aim to identify a set $S \subseteq V$ that maximizes $f(S) - c(S)$, where $c(S)$ denotes the sum of the costs of the elements in S.

We refer to this problem as *unconstrained submodular maximization with modular cost (USM-MC)*, and refer to $f(S) - c(S)$ as the *profit* of S. USM-MC finds applications in numerous practical scenarios, such as *profit maximization* [25, 33], *sensor placement*, and *text summarization*.

In particular, in profit maximization [25, 33], we are given a set V of social network users, each of which can be incentivized at a cost to be an initiator in a viral marketing campaign. Our objective is to select a set $S \subseteq V$ of initiators that maximizes $f(S) - c(S)$, where $f(S)$ is the expected revenue resulted from the marketing campaign, and $c(S)$ is the total incentive paid to the users in S. In sensor placement, we have a set V of locations at which sensors can be placed, and a function f (resp. c) that, given a set $S \subseteq V$, returns the financial benefit (resp. total cost) of installing sensors at the locations in S. In this setting, a set S that maximizes $f(S) - c(S)$ is an ideal option for sensor placement with balanced cost and benefit. In text summarization, we are to select a set S of words to summarize a given set of documents, based on a function $f(S)$ that evaluates the summarization accuracy of S and a function $c(S)$ that returns the total length of the words in S. Assuming that $f(S)$ and $c(S)$ are properly normalized, we can select a set S that maximizes $f(S) - c(S)$ to strike a good trade-off between the accuracy and length of the summarization.

Motivation. As we show in Section 3.1, USM-MC is an NP-hard problem. Existing work [25, 37] on USM-MC has mostly relied on heuristics without non-trivial worst-case guarantees, with only two exceptions: Double-Greedy [13, 33, 34] and Distorted-Greedy [12, 16]. However, both of them have significant limitations. Specifically, Double-Greedy [4] returns a set S whose profit satisfies

$$f(S) - c(S) \geq 1/3(f(S^*) - c(S^*)), \text{ or}$$
$$\mathbb{E}\left[f(S) - c(S)\right] \geq 1/2(f(S^*) - c(S^*)),$$

*Corresponding author

Proceedings of the VLDB Endowment, Vol. 14, No. 10
ISSN 2150-8097.
doi:10.14778/3467861.3467866

where S^* is the optimal solution to USM-MC, and the expectation in the second inequality is taken over the randomness in Double-Greedy. Nevertheless, this profit guarantee holds only when $f(S') - c(S') \geq 0$ is satisfied for every $S' \subseteq V$, which is seldom the case in practice. For example, in viral marketing, we usually have $f(S') - c(S') < 0$ when $S' = V$, as it is unprofitable to pay all social network users to be initiators. In contrast, Distorted-Greedy [12] does not rely on any unrealistic assumption on f or c. However, the approximation guarantee of Distorted-Greedy is relatively weak: it returns a set S such that

$$f(S) - c(S) \geq \left(1 - \frac{1}{e}\right) f(S^*) - c(S^*) = f(S^*) - c(S^*) - \frac{f(S^*)}{e \cdot c(S^*)} \cdot c(S^*). \quad (1)$$

In other words, compared with the optimal solution S^*, Distorted-Greedy pays an extra cost linear to $f(S^*)$ in the worst case, which is rather unfavorable when $f(S^*)$ is large. Further, when $\frac{f(S^*)}{c(S^*)} \leq \frac{1}{1-1/e} \approx 1.58$, Distorted-Greedy does not even guarantee a positive profit. As such, if we apply Distorted-Greedy to solve the profit maximization problem, then it may not be able to identify a profitable solution, even when the optimal solution S^* has an expected revenue $f(S^*)$ that is 50% larger than its cost $c(S^*)$.

Contributions. To address the deficiencies of Double-Greedy and Distorted-Greedy, this paper presents ROI-Greedy, a polynomial-time algorithm for USM-MC that returns a set S such that

$$f(S) - c(S) \geq f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*). \quad (2)$$

That is, compared with the optimal solution S^*, ROI-Greedy incurs an extra cost that is asymptotically much smaller than that of Distorted-Greedy (see Eq. (1)), and is quantitatively never larger than the latter. Further, the r.h.s. of Eq. (2) is positive whenever $f(S^*) > c(S^*)$, *i.e.*, ROI-Greedy yields a positive profit as long as the optimal solution S^* is profitable. This is a significant improvement over the profit guarantee of Distorted-Greedy. In addition, ROI-Greedy does not make any strong assumptions on f or c.

ROI-Greedy is based on a simple idea: we start from an empty solution set S, and iteratively add elements in V into S, until none of the remaining elements can be added without degrading the profit of S. In particular, in each iteration, we insert into S the element $u \in V \setminus S$ that maximizes $\frac{\Delta(u|S)}{c(\{u\})}$, where $\Delta(u \mid S) = f(S \cup \{u\}) - f(S)$ denotes the marginal increase in $f(S)$ when u is added into S. In other words, we greedily choose u to maximize the *return-on-investment (ROI)* from u. This approach is similar in spirit to the greedy approximation algorithm for the knapsack problem[1] [2, 18], but establishing its theoretical guarantee for USM-MC is challenging due to a crucial difference between knapsack and USM-MC: the objective function of knapsack is monotone, whereas in USM-MC, the objective function $f(S) - c(S)$ is non-monotone with respect to S. We address this challenge and prove Eq. (2) with a series of careful analysis, as shown in Section 3. In addition, we prove that the profit guarantee in Eq. (2) is *tight*, in the sense that no polynomial-time algorithm can ensure

$$f(S) - c(S) \geq (1 + \epsilon) \left(f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)\right), \quad (3)$$

for any $\epsilon > 0$, unless P = NP.

Table 1: Frequently used notations.

Notation	Description
V	a set of n elements
$f(\cdot)$	a monotone, non-negative, and submodular function
$c(\cdot)$	a non-negative, and modular function
$\Delta(u \mid S)$	the marginal gain of adding element u to S
S°	the output of ROI-Greedy
S^*	the optimal solution to USM-MC
$G = (V, E)$	a graph G with nodes V and edges E
n, m	$n = \|V\|, m = \|E\|$
$\mathcal{R}$	a set of RR sets
$Cov_{\mathcal{R}}(S)$	the number of RR sets covered by S
$f_{\mathcal{R}}(S)$	$f_{\mathcal{R}}(S) = n \cdot Cov_{\mathcal{R}}(S)/\|\mathcal{R}\|$

Our technical contributions are summarized as follows.

- We present ROI-Greedy, a polynomial-time algorithm for the USM-MC problem. ROI-Greedy returns results satisfying a *strong* worst-case approximation guarantee as shown in Eq. (2), which, to our knowledge, was never achieved before.
- We prove that the worst-case guarantee is *tight*, unless P=NP, as shown in Eq. (3).
- In practice, we develop a non-trivial extension of ROI-Greedy to solve the profit maximization problem and conduct extensive experiments, demonstrating that ROI-Greedy significantly outperforms competitors in terms of the trade-off between efficiency and result quality.

2 UNCONSTRAINED SUBMODULAR MAXIMIZATION WITH MODULAR COSTS

2.1 Problem Definition and Notations

Let V be a set of size n, and $f : 2^V \to \mathbb{R}^+ \cup \{0\}$ be a *score* function that is *monotone* and *submodular*, *i.e.*, for any two sets $A \subseteq B \subseteq V$,

$$f(A) \leq f(B), \quad \text{(monotonicity)}$$
$$\text{and } f(\{u\} \cup A) - f(A) \geq f(\{u\} \cup B) - f(B). \quad \text{(submodularity)}$$

For convenience, we define $\Delta(u \mid S) = f(\{u\} \cup S) - f(S)$ for any $u \in V$ and any $S \subseteq V$.

Assume that each $u \in V$ is associated with a positive cost c_u, and let $c(S) = \sum_{u \in S} c_u$. Given V, f, and c, the problem of *unconstrained submodular maximization with modular costs (USM-MC)* asks for a set $S \subseteq V$ that maximizes $f(S) - c(S)$, referred to as the *profit* of S. Table 1 lists the notations that are frequently used in this paper.

2.2 State of the Art

Existing solutions [4, 12, 13, 16, 25, 33, 34, 37] for USM-MC are based on three greedy approaches, as we explain as follows.

Simple-Greedy [25, 37]. Simple-Greedy starts from $S = \emptyset$, and iteratively inserts into S the element u in $V \setminus S$ that maximizes the *marginal profit gain*, defined as $\Delta(u \mid S) - c_u$. This iterative process terminates when none of the elements in $V \setminus S$ has a positive

[1] Given V, f, c, and a threshold τ, the knapsack problem asks for a set $S \subseteq V$ that maximizes $f(S)$ subject to the constraint that $c(S) \leq \tau$.

marginal profit gain. The resulting set S is returned as the output of Simple-Greedy. Simple-Greedy is intuitive, but it fails to provide any non-trivial guarantee in terms of the profit of S. To demonstrate this, we illustrate an example where Simple-Greedy yields a profit that is arbitrarily worse than the optimal solution, and the gap between these two is $O(n)$.

Example 1. Assume that $V = \{u_1, u_2, \ldots, u_n\}$, such that:

- $c_{u_n} = n - 1$, while $c_{u_i} = 0.6$ for any $i \in [1, n-1]$;
- for any $S \subseteq V \setminus \{u_n\}$, $f(S) = |S|$;
- for any $S \subseteq V$ that contains u_n, $f(S) = n$.

Suppose that we apply Simple-Greedy on V. It starts with $S = \emptyset$, and then inserts u_n into S in the first iteration, since u_n's marginal profit gain is $\Delta(u_n \mid \emptyset) - c_{u_n} = 1$, while the marginal profit gain of any other element is 0.6. Once u_n is inserted into S, the marginal profit gain of every remaining element becomes negative, and hence, Simple-Greedy terminates and returns $S = \{u_n\}$, which has a profit of 1. Meanwhile, the optimal solution is $S^* = \{u_1, u_2, \ldots, u_{n-1}\}$, which has a profit of $0.4 \cdot (n-1)$. That is, Simple-Greedy achieves a profit that is $\frac{1}{0.4 \cdot (n-1)}$ times the optimum. □

Double-Greedy [4]. Similar to Simple-Greedy, Double-Greedy also constructs a solution S by starting from $S = \emptyset$ and iteratively inserting elements into S. However, it does not follow the greedy framework in Simple-Greedy; instead, it maintains an auxiliary set T, and uses T in deciding if an element should be added into S.

Specifically, Double-Greedy starts with $S = \emptyset$ and $T = V$. After that, it linearly scans the elements in T in an arbitrary order. For each element u examined, Double-Greedy evaluates its marginal profit gain $\Delta(u \mid S) - c_u$ with respect to S, as well as its *reverse* marginal profit gain with respect to T, defined as $c_u - \Delta(u \mid T \setminus \{u\})$. If it turns out that

$$\Delta(u \mid S) - c_u > c_u - \Delta(u \mid T \setminus \{u\}),$$

then Double-Greedy inserts u into S before proceeding to the next element; otherwise, it removes u from T. In other words, u is either (i) inserted into S and retained in T, or (ii) removed from T without being added into S. Once all elements in T are inspected, Double-Greedy returns the set S constructed.

When $f(S') - c(S') \geq 0$ for every $S' \subseteq V$, Double-Greedy ensures that

$$f(S) - c(S) \geq 1/3(f(S^*) - c(S^*)).$$

In addition, by linearly scanning the elements in T in a carefully randomized order, Double-Greedy also guarantees that

$$\mathbb{E}\left[f(S) - c(S)\right] \geq 1/2(f(S^*) - c(S^*)),$$

where the expectation is taken over the randomness in ordering elements. However, as we mentioned in Section 1, it is unrealistic to assume $f(S') - c(S') \geq 0$ for all $S' \subseteq V$ in practice.

Distorted-Greedy [12]. Distorted-Greedy is similar to Simple-Greedy in that they both start from $S = \emptyset$ and iteratively insert elements into S until a termination condition is satisfied. There are only two differences. First, in the i-th ($i = 1, 2, \ldots$) iteration of Distorted-Greedy, it selects the element $u_i \in V \setminus S$ that maximizes the *distorted* marginal profit gain, which is defined as

$$\left(1 - \tfrac{1}{n}\right)^{n-i} \cdot \Delta(u \mid S) - c_u.$$

Second, Distorted-Greedy terminates when none of the elements in $V \setminus S$ has a positive distorted marginal profit gain. It is shown [12] that Distorted-Greedy returns a set S such that

$$f(S) - c(S) \geq \left(1 - \tfrac{1}{e}\right) f(S^*) - c(S^*). \tag{4}$$

This profit guarantee, however, is relatively weak for two reasons, as mentioned in Section 1: (i) the profit gap between S and S^* increases linearly with $f(S^*)$, and (ii) Distorted-Greedy does not guarantee a positive profit whenever $\frac{f(S^*)}{c(S^*)} < \frac{1}{1-1/e} \approx 1.58$.

3 ROI-Greedy

This section presents our ROI-Greedy algorithm for the USM-MC problem. We first prove that USM-MC is NP-hard in Section 3.1, and then present the details of ROI-Greedy in Section 3.2. We establish the approximation guarantee of ROI-Greedy in Section 3.3, and prove the tightness of the approximation in Section 3.4.

3.1 NP-Hardness of USM-MC

We prove the NP-hardness of USM-MC by a reduction from the *minimum set cover* problem [10]. Let $V = \{u_1, u_2, \ldots, u_n\}$ be a set such that each u_i is a set of items, and let $N = |\cup_{i=1}^{n} u_i|$. A set $S \subseteq V$ is a *set cover* for V, if $|\cup_{u_i \in S} u_i| = N$. The minimum set cover problem asks for a set cover S such that $|S|$ is minimized.

Given an instance V of the minimum set cover problem, we construct an instance of USM-MC using V as follows. First, for any $S \subseteq V$, we set $f(S) = |\cup_{u_i \in S} u_i|$. In addition, we set $c_{u_i} = \frac{1}{2}$ for any u_i. Then, f is non-negative, monotone, and submodular, while c is non-negative and modular. Then, the set $S^* \subseteq V$ that maximizes $f(S^*) - c(S^*)$ must have $f(S^*) = N$; otherwise, we could increase $f(S^*) - c(S^*)$ by inserting into S^* any $u_i \notin \cup_{u \in S^*} u$. Therefore, S^* must be a set cover for V. In addition, among all $S \subseteq V$ such that $|\cup_{u_i \in S} u_i| = N$, S^* must have the smallest size; otherwise, there exists another set $S' \subseteq V$ such that $f(S') = f(S^*) = N$ and $c(S') < c(S^*)$, leading to a contradiction. Therefore, S^* is a minimum set cover for V. This completes the proof.

3.2 Our Algorithm

Algorithm 1 shows the pseudo-code of our ROI-Greedy algorithm. ROI-Greedy adopts the same greedy framework used by Simple-Greedy and Distorted-Greedy, *i.e.*, it constructs its solution set S° by iteratively inserting elements into an initially empty set. Its main difference from Simple-Greedy and Distorted-Greedy lies in the metric that it uses to select the element to be inserted: in the i-th iteration, it chooses the element $v_i \in V \setminus S_{i-1}$ that maximizes $\frac{\Delta(v_i \mid S_{i-1})}{c_{v_i}}$, where S_{i-1} denotes the solution set constructed in the first $i-1$ iterations (Line 5 of Algorithm 1). The intuition for this choice of v_i is that it has the best "return on investment (ROI)" in terms of the cost that we pay to add v_i into our solution set. If $\frac{\Delta(v_i \mid S_{i-1})}{c_{v_i}} > 1$, then ROI-Greedy inserts v_i into S_{i-1} and proceeds to the next iteration (Lines 6-7); otherwise, it terminates and returns $S^\circ = S_{i-1}$ as the solution (Lines 9-10). We show that S° provides a strong profit guarantee as follows:

Algorithm 1: ROI-Greedy

Input: Set V and functions $f(\cdot)$ and $c(\cdot)$.
Output: S°.

$S_0 \leftarrow \emptyset$;
$i \leftarrow 0$;
while $i < n$ **do**
  $i \leftarrow i + 1$;
  $v_i \leftarrow \arg\max_{u \in V \setminus S_{i-1}} \frac{\Delta(u|S_{i-1})}{c_u}$;
  if $\Delta(v_i \mid S_{i-1}) - c_{v_i} > 0$ **then**
    $S_i \leftarrow \{v_i\} \cup S_{i-1}$;
  else
    $S^\circ \leftarrow S_{i-1}$;
    return S°;
$S^\circ \leftarrow V$;
return S°;

Theorem 1. *Let S° be the output of Algorithm 1. Then,*

$$f(S^\circ) - c(S^\circ) \geq \max_{S \subseteq V} \left(f(S) - c(S) - \ln \frac{f(S)}{c(S)} \cdot c(S) \right).$$

We present the proof of Theorem 1 in Section 3.3. As an immediate corollary, we have[2]:

Corollary 1. *$f(S^\circ) - c(S^\circ) \geq f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)$, where S^* is the optimal solution to USM-MC.*

Comparison of Approximation Guarantees. Recall that Distorted-Greedy [12] returns a solution S with

$$f(S) - c(S) \geq f(S^*) - c(S^*) - \tfrac{1}{e} f(S^*).$$

To compare this profit guarantee with that of ROI-Greedy in Corollary 1, we first observe that when $\frac{f(S^*)}{c(S^*)} \leq \frac{1}{1-1/e}$, $f(S^*) - c(S^*) - \frac{1}{e} f(S^*) < 0$, in which case Distorted-Greedy does not guarantee a positive profit. In contrast, when $f(S^*) - c(S^*) > 0$,

$$f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*) = \left(\frac{f(S^*)}{c(S^*)} - 1 - \ln \frac{f(S^*)}{c(S^*)} \right) \cdot c(S^*) > 0,$$

since the function $g(x) = x - 1 - \ln x$ is positive when $x > 1$. In other words, ROI-Greedy always yields a positive profit when the optimal solution S^* is profitable.

In addition, observe that

$$\left(f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*) \right) - \left(f(S^*) - c(S^*) - \tfrac{1}{e} f(S^*) \right)$$
$$= \tfrac{1}{e} f(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)$$
$$= \left(\tfrac{1}{e} \frac{f(S^*)}{c(S^*)} - \ln \frac{f(S^*)}{c(S^*)} \right) \cdot c(S^*),$$

it can be verified that $\frac{1}{e} \frac{f(S^*)}{c(S^*)} - \ln \frac{f(S^*)}{c(S^*)}$ is always non-negative, and it increases monotonically with $\frac{f(S^*)}{c(S^*)}$ when $\frac{f(S^*)}{c(S^*)} \geq e$. This shows that the approximation guarantee of ROI-Greedy is no worse than that of Distorted-Greedy, and is significantly better than the latter when $\frac{f(S^*)}{c(S^*)}$ is large.

Compared with the approximation guarantee of Double-Greedy, ROI-Greedy's approximation guarantee is much stronger in the sense that it does not rely on any strong assumption on f or c, whereas Double-Greedy requires that $f(S') - c(S') \geq 0$ for all $S' \subseteq V$. It can also be verified that ROI-Greedy's profit guarantee is higher than that of Double-Greedy when $\frac{f(S^*)}{c(S^*)}$ is large; the detailed analysis for this is omitted for brevity.

3.3 Proof of Theorem 1

Let $S_0 = \emptyset$, and S_{i-1} ($i > 1$) be the partial solution set constructed by the first $i-1$ iterations of ROI-Greedy. Let S^+ be the subset of V that maximizes $f(S) - c(S) - \ln \frac{f(S)}{c(S)} \cdot c(S)$. Our proof of Theorem 1 utilizes two lemmas as follows [3].

Lemma 1. $\Delta(v_i \mid S_{i-1}) \geq \frac{c_{v_i}}{c(S^+)} \left(f(S^+) - f(S_{i-1}) \right)$.

Lemma 2.

$$f(S_i) \geq \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{v_k}}{c(S^+)} \right) \right) \cdot f(S^+). \tag{5}$$

We also utilize a classic inequality [2] as follows: For any $x_1, x_2, \cdots, x_n, y \in \mathbb{R}^+$ and $x_i \leq y$ for $i \leq n$,

$$1 - \prod_{i=1}^{n} \left(1 - \frac{x_i}{y} \right) \geq 1 - \left(1 - \frac{\sum_{i=1}^{n} x_i}{ny} \right)^n. \tag{6}$$

Based on Lemmas 1, 2 and Eq. (6), we will prove Theorem 1 by showing that

$$f(S^\circ) - c(S^\circ) \geq \left(f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+) \right). \tag{7}$$

We consider two cases based on whether $c(S^\circ) < \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+)$.

Case 1: $c(S^\circ) < \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+)$. In this case, by Algorithm 1,

$$0 > \max_{u \in V \setminus S^\circ} (\Delta(u \mid S^\circ) - c_u) \geq \max_{u \in S^+ \setminus S^\circ} (\Delta(u \mid S^\circ) - c_u)$$
$$\geq \Delta(S^+ \mid S^\circ) - c(S^+) \qquad \geq f(S^+) - f(S^\circ) - c(S^+).$$

Hence, $f(S^\circ) \geq f(S^+) - c(S^+)$. This leads to

$$f(S^\circ) - c(S^\circ) \geq f(S^+) - c(S^+) - c(S^\circ)$$
$$\geq f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} c(S^+).$$

Case 2: $c(S^\circ) \geq \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+)$. In this case, there exists $t \in [1, n]$ such that $S_{t-1} \subseteq S_t \subseteq S^\circ$ and $c(S_t) \geq \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+) > c(S_{t-1})$. Observe that if $f(S_{t-1}) \geq f(S^+)$, then Eq. (7) trivially holds. In addition, if $f(S_{t-1}) < f(S^+)$ and

$$f(S_{t-1}) - c(S_{t-1}) \geq f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+),$$

then Eq. (7) also holds. Thus, in what follows, we only consider that

$$f(S_{t-1}) < f(S^+) \qquad \text{and} \tag{8}$$

$$f(S_{t-1}) - c(S_{t-1}) < f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+). \tag{9}$$

[2] Note that in Corollary 1, the r.h.s. of the inequality is no larger than $f(S^*) - c(S^*)$. This is because (i) $c(S^*) > 0$ and (ii) $\ln \frac{f(S^*)}{c(S^*)} \geq 0$, since $f(S^*) - c(S^*) \geq f(\emptyset) - c(\emptyset) = f(\emptyset) \geq 0$.

[3] All the omitted proofs can be found in the appendix

Let $\beta = \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+) - c(S_{t-1})$. We have the following lemma.

Lemma 3.

$$\frac{\beta(f(S^+) - f(S_{t-1}))}{c(S^+)} + f(S_{t-1}) \geq f(S^+) - c(S^+). \tag{10}$$

Given Lemma 3, we have the following results. First,

$$\begin{aligned}
&\frac{\beta(f(S^+) - f(S_{t-1}))}{c(S^+)} + f(S_{t-1}) - c(S_{t-1}) - \beta \\
&\geq f(S^+) - c(S^+) - c(S_{t-1}) - \beta \\
&= f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+) \\
&\geq f(S_{t-1}) - c(S_{t-1}),
\end{aligned} \tag{11}$$

where the first inequality is due to Eq. (10) and the last inequality is due to Eq. (9). By Eq. (11),

$$\frac{\beta(f(S^+) - f(S_{t-1}))}{c(S^+)} - \beta \geq 0.$$

Hence,

$$f(S^+) - f(S_{t-1}) \geq c(S^+). \tag{12}$$

Finally, we have

$$\begin{aligned}
&f(S^\circ) - c(S^\circ) \quad \geq f(S_t) - c(S_t) \\
&= \Delta(v_t \mid S_{t-1}) + f(S_{t-1}) - c(S_t) \\
&\geq \frac{c_{v_t}(f(S^+) - f(S_{t-1}))}{c(S^+)} + f(S_{t-1}) - c(S_{t-1}) - c_{v_t} \\
&= \frac{(c_{v_t} + \beta - \beta)(f(S^+) - f(S_{t-1}))}{c(S^+)} + f(S_{t-1}) \\
&\qquad - \ln \frac{f(S^+)}{c(S^+)} c(S^+) - (c_{v_t} - \beta) \\
&\geq f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} c(S^+) + (c_{v_t} - \beta)\left(\frac{f(S^+) - f(S_{t-1})}{c(S^+)} - 1\right) \\
&\geq f(S^+) - c(S^+) - \ln \frac{f(S^+)}{c(S^+)} c(S^+),
\end{aligned}$$

where the third inequality is from Eq. (10), and the last inequality is due to Eq. (12) and $c_{v_t} = c(S_t) - c(S_{t-1}) \geq \beta$.

3.4 Tightness of Approximation

In what follows, we show that the approximation guarantee of ROI-Greedy in Corollary 1 is tight, in the sense that no polynomial time algorithm can improve it by a multiplicative factor.

Theorem 2. *Assume that* P≠NP. *Then, for any $\epsilon > 0$, there does not exist a polynomial-time algorithm whose output S' guarantees*

$$f(S') - c(S') \geq (1+\epsilon)\left(f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)\right). \tag{13}$$

where S^* is the optimal solution to USM-MC.

4 PROFIT MAXIMIZATION

In this section, we extend ROI-Greedy to address the *profit maximization* problem, and obtain an approximation guarantee that is significantly better than those of all existing methods. Section 4.1 defines the profit maximization problem. Section 4.2 presents our algorithm, while Section 4.3 provides our theoretical analysis.

4.1 Problem Definition

Let $G = \langle V, E \rangle$ be a social network with a set V of nodes and a set E of edges. Let $n = |V|$ and $m = |E|$. We consider an *influence diffusion* process in G that starts from a set S of *seed nodes* as follows:

(1) At timestamp 1, the nodes in S are *activated*, while the remaining nodes are inactive.
(2) At any subsequent timestamp t ($t > 1$), each node v that was newly activated in timestamp $t-1$ has a chance to *activate* its neighbors, based on a certain *diffusion model*.
(3) If no new node is activated in a timestamp, then the diffusion process terminates.

Let $I(S)$ be the set of nodes that are activated at the end of the influence diffusion process. We say that the nodes in $I(S)$ are *influenced* by S, and refer to $|I(S)|$ as the *spread* of S.

Let $f(S) = \mathbb{E}[|I(S)|]$, where the expectation is taken over the randomness of the diffusion process. For each node $u \in V$, let $c_u > 0$ be the incentive needed to convince u to be a seed node, and let $c(S) = \sum_{u \in S} c_u$. The *profit maximization* problem asks for a seed set S with the maximum *profit*, defined as $f(S) - c(S)$.

We study the profit maximization problem when the diffusion model used is *independent cascade (IC)* [17], as it is the most well-adopted model in the literature. In the IC model, each edge (u, v) in E is associated with a propagation probability $p(u, v) \in [0, 1]$. If u is activated at timestamp $t-1$, then at the next timestamp t, u can activate v with a probability $p(u, v)$, independent of the activation of other neighbors of u. It is shown that, under the IC model, the function $f(S) = \mathbb{E}[|I(S)|]$ is both monotone and submodular [17]. Therefore, under the IC model, profit maximization is an instance of our USM-MC problem.

4.2 Our Algorithm

ROI-Greedy cannot be directly applied on profit maximization under the IC model, since it requires assessing $f(S) = \mathbb{E}[|I(S)|]$ for some $S \subseteq V$, whereas computing the exact expected spread $\mathbb{E}[|I(S)|]$ of S is #P-hard in general [6]. To address this issue, we estimate $f(S)$ with sampling, and revise ROI-Greedy to account for the estimation error in $f(S)$.

In particular, we estimate $f(S)$ using *reverse influence sampling (RIS)* [3], which is also adopted in the state of the art [13, 34] for profit maximization. Each sample R from RIS, referred to a *reverse reachable (RR)* set, is a subset of V conceptually generated as follows:

(1) Consider a random graph G' generated from G by retaining all nodes in G but removing each edge (u, v) from G independently with probability $1 - p(u, v)$.
(2) Choose a node v uniformly at random from G'.
(3) Let R be the set of all nodes u such that there exists a directed path in G' that starts from u and ends at v.

We say that an RR set R is *covered* by a seed set S if $R \cap S \neq \emptyset$. Borgs *et al.* [3] prove that

$$f(S) = n \cdot \mathbb{P}[R \cap S \neq \emptyset].$$

Given a set $\mathcal{R} = \{R_1, R_2, \ldots\}$ of RR sets, we use $Cov_{\mathcal{R}}(S)$ to denote the number of RR sets in $\mathcal{R}$ that are covered by S. Let

$$f_{\mathcal{R}}(S) = n \cdot \frac{Cov_{\mathcal{R}}(S)}{|\mathcal{R}|}.$$

Then, when $\mathcal{R}$ is fixed, $f_{\mathcal{R}}(S)$ is an unbiased estimation of $f(S)$. In other words, $\mathcal{R}$ can be used as a *noisy* oracle that returns an estimated $f(S)$ for any given $S \subseteq V$.

Based on ROI-Greedy and RIS, we propose ROI-PM, an efficient algorithm for profit maximization that, with at least $1-\delta$ probability, returns a solution S° satisfying

$$f(S^\circ) - c(S^\circ) \geq (1-\epsilon)f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*), \quad (14)$$

where S^* is the optimal solution and $\epsilon, \delta \in (0,1)$ are user-defined parameters. Algorithm 2 shows the pseudo-code of ROI-PM, while Algorithm 3 illustrates a sub-routine invoked in Line 4 of Algorithm 2. The basic idea of ROI-PM is the same as ROI-Greedy's: it starts from an empty solution set $S^\circ = \emptyset$ and interatively inserts nodes into S°, attempting to ensure that each inserted node u maximizes $\frac{\Delta(u|S^\circ)}{c_u}$. However, since the exact value of $\Delta(u \mid S^\circ) = f(\{u\} \cup S^\circ) - f(S^\circ)$ cannot be computed in polynomial time, ROI-PM has to resort to an estimation of $\Delta(u \mid S^\circ)$ via RIS-based sampling. This leads to a challenging question: how large a sample set should we use to achieve the profit guarantee in Eq. (14) without excessive computation overheads? ROI-PM addresses this challenge using a *trial-and-error* approach as follows.

First, ROI-PM generates a relatively small sample set $\mathcal{R}_1$ of size n (Lines 1-3 in Algorithm 2). Then, using $\mathcal{R}_1$ as a noisy oracle for estimating $f(\cdot)$, it employs the greedy approach in ROI-Greedy to generate a solution S° to the profit maximization problem (Line 4 in Algorithm 2). After that, it uses another sample set $\mathcal{R}_2$ (with $|\mathcal{R}_2| = |\mathcal{R}_1|$) as an *independent oracle* to verify whether S° is a high-quality solution (Lines 6-9). The intuition is that if we compute an estimation of $f(S^\circ)$ from $\mathcal{R}_2$ and it turns out to be much smaller than the estimation derived from $\mathcal{R}_1$, and it could be the case that $\mathcal{R}_1$ over-estimates S°'s spread. In that case, S° might not be a good solution; accordingly, ROI-PM would discard S°, double the sizes of the sample sets $\mathcal{R}_1$ and $\mathcal{R}_2$ (Line 10), and then regenerate another S° from scratch using the updated $\mathcal{R}_1$ as a noisy oracle. This process is repeated until (i) the independent oracle $\mathcal{R}_2$ agrees with the noisy oracle $\mathcal{R}_1$ on the quality of S° (Lines 8-9), or (ii) the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ reach a pre-defined threshold (Line 11). After that, ROI-PM terminates by returning S° (Line 12).

Remark. Our solution can be extended to address profit maximization under other diffusion models, as long as (i) $\mathbb{E}[|I(S)|]$ is monotone and submodular, and (ii) $\mathbb{E}[|I(S)|]$ can be computed by reverse influence sampling (RIS). For instance, ROI-PM can be extended to the linear threshold model [17] since it is monotone and submodular, and the influence in the linear threshold model can be computed by RIS [35]. Similarly, ROI-PM can also be extended to the triggering model [17].

4.3 Theoretical Analysis

In the following, we analyze the approximation guarantee and time complexity of ROI-PM. To facilitate our analysis, we first introduce Chernoff Inequalities [27] in Lemma 4, followed by a result in Lemma 5 that we frequently used.

Lemma 4 (Chernoff Inequalities [27]). *Let $X_1, \ldots, X_M$ be independent random variables in $[0,1]$. Let $X = \sum_{i=1}^{M} X_i$ and $\mu = \mathbb{E}[X_i]$. Then, for any $\lambda > 0$,*

$$\mathbb{P}[X - M\mu \geq \lambda \cdot M\mu] \leq \exp\left(-\frac{\lambda^2}{2+\lambda} M\mu\right), \text{ and}$$

$$\mathbb{P}[X - M\mu \leq -\lambda \cdot M\mu] \leq \exp\left(-\frac{\lambda^2}{2} M\mu\right).$$

Algorithm 2: ROI Profit Maximization (ROI-PM)

Input: Graph G; Costs of each node $\{c_v\}_{v \in V}$; Parameter ϵ and δ.
Output: S°.

1 $S^\circ \leftarrow \emptyset$, $\theta_1 \leftarrow n$, $i \leftarrow 1$;
2 **do**
3 Generate two collections of RR sets $\mathcal{R}_1$ and $\mathcal{R}_2$, where $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_i$;
4 $S^\circ \leftarrow$ NodeSelection($\mathcal{R}_1$);
5 $t \leftarrow (f_{\mathcal{R}_1}(S^\circ) - c(S^\circ))/(f_{\mathcal{R}_2}(S^\circ) - c(S^\circ))$;
6 Let ϵ_1 be the larger root of $(\epsilon_1+1)(\epsilon_1+2)/\epsilon_1^2 = f_{\mathcal{R}_2}(S^\circ)/\ln(6 \cdot i^2/\delta) \cdot \theta_i/n$;
7 Let ϵ_2 be the larger root of $(2\epsilon_1+2)/\epsilon_2^2 = (f_{\mathcal{R}_2}(S^\circ) - (1+\epsilon_1)c(S^\circ))/\ln(6 \cdot i^2/\delta) \cdot \theta_i/n$;
8 **if** $(t-1)/t + \epsilon_2 + \epsilon_1 \leq \epsilon$, $\epsilon_1 + \epsilon_2 \leq \epsilon$, *and* $t, \epsilon_1, \epsilon_2 > 0$ **then**
9 **Break;**
10 $i \leftarrow i+1$, $\theta_i \leftarrow 2\theta_i$;
11 **while** $\theta_i \leq (8+2\epsilon)(1+\epsilon_1)n \frac{\ln(6/\delta)+n\ln 2}{\epsilon^2 \max\{1, f_{\mathcal{R}_2}(S^\circ)-(1+\epsilon_1)c(S^\circ)\}}$;
12 **return** S°

Lemma 5. *Suppose that $c(S_1) = c(S_2)$, $f(S_1) \geq f(S_2) - x$ for some $x \in (0,1)$, and $f(S_2) - c(S_2) \geq 0$. Then,*

$$f(S_1) - c(S_1) - \ln \frac{f(S_1)}{c(S_1)} \cdot c(S_1) \geq f(S_2) - x - c(S_2) - \ln \frac{f(S_2)}{c(S_2)} \cdot c(S_2)$$

Proof.

$$\begin{aligned} & f(S_1) - c(S_1) - \ln \frac{f(S_1)}{c(S_1)} \cdot c(S_1) \\ & \geq f(S_1) - \frac{c(S_2)}{f(S_2)} \cdot f(S_1) - \ln \frac{f(S_2)}{c(S_2)} \cdot c(S_1) \quad (15) \\ & \geq (1 - \frac{c(S_2)}{f(S_2)})(f(S_2) - x) - \ln \frac{f(S_2)}{c(S_2)} \cdot c(S_2) \\ & \geq f(S_2) - x - c(S_2) - \ln \frac{f(S_2)}{c(S_2)} \cdot c(S_2), \end{aligned}$$

where the first inequality holds because (i) the function $g(y) = (1-e^{-y})f(S_1) - y \cdot c(S_1)$ achieves its minimum at $y = \ln \frac{f(S_1)}{c(S_1)}$; (ii) $g\left(\ln \frac{f(S_1)}{c(S_1)}\right)$ equals the l.h.s. of Eq. (15); (iii) $g\left(\ln \frac{f(S_2)}{c(S_2)}\right)$ equals the r.h.s. of Eq. (15). □

Based on Lemma 4, we prove that with a high probability, in each round of Algorithm 2 (Lines 3-10), $f_{\mathcal{R}_2}(S^\circ)$ is not a significant overestimation of $f(S^\circ)$, while $f_{\mathcal{R}_1}(S^*)$ is not a significant underestimation of $f(S^*)$.

Lemma 6. *With probability at least $1 - \frac{5\delta}{9}$, we have*

$$f_{\mathcal{R}_2}(S^\circ) \leq (1+\epsilon_1) f(S^\circ), \text{ and} \quad (16)$$

$$f_{\mathcal{R}_1}(S^*) \geq (1-\epsilon_2) f(S^*) \quad (17)$$

for every iteration (Line 3~Line 10) of Algorithm 2 when $\epsilon_1, \epsilon_2, t > 0$.

Algorithm 3: NodeSelection

Input: RR sets $\mathcal{R}$.
Output: S°.

```
S° ← ∅;
Let Cov_R(S) be the number of RR sets covered by S in R;
Let r ← |R|;
while true do
    v ← arg max_{u∈V} (n · Cov_R(u)/(r · c_u)) ;
    if n · Cov_R(v)/r − c_v ≤ 0 then
        Break;
    Add v to S°;
    Remove RR sets covered by v;
return S°;
```

Table 2: Dataset statistics (K = 10^3, M = 10^6).

Name	n	m	Type	Average degree
NetHEPT	15.2K	31.4K	undirected	4.18
Epinions	132K	841K	directed	13.4
DBLP	655K	1.99M	undirected	6.08
LiveJournal	4.85M	69.0M	directed	28.5

Based on Lemma 6, we prove the approximation guarantee of ROI-PM as follows.

Theorem 3 (Approximation Guarantee of ROI-PM). *With probability at least $1-\delta$, the solution S° returned by Algorithm 2 satisfies*

$$f(S^\circ) - c(S^\circ) \geq (1-\epsilon) f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*), \quad (18)$$

where S^* is the optimal solution.

The following theorem shows the time complexity of ROI-PM.

Theorem 4. *The expected time complexity of Algorithm 2 is $O\left(\frac{m \cdot f(\{v^*\}) \cdot \max\{\ln \delta^{-1}, n\}}{\epsilon^2}\right)$, where v^* is the node in G with the largest expected spread.*

Proof. The computation overhead of Algorithm 2 is dominated by the cost of RR set generation. By Line 11 of Algorithm 2, the total number of RR sets generated is at most $O(n \cdot \max\{\ln \delta^{-1}, n\}/\epsilon^2)$. As shown in [35], the expected cost of generating a random RR set is bounded by $\frac{m}{n} \cdot f(\{v^*\})$. Therefore, the expected time complexity of Algorithm 2 is $O\left(\frac{m \cdot f(\{v^*\}) \cdot \max\{\ln \delta^{-1}, n\}}{\epsilon^2}\right)$. □

5 EXPERIMENTS

In this section, we experimentally evaluate the proposed algorithms against existing state-of-the-art solutions in terms of both effectiveness and efficiency on real-world graphs[4]. All experiments are conducted on a Linux server with an Intel Xeon 2.60GHz CPU and 376GB RAM. All algorithms are implemented in C++ and compiled by g++ 7.4 with -O3 optimization.

[4]The experiments on sensor placement and text summarization could be found in https://sites.google.com/view/roi-greedy-tr.

5.1 Experimental Settings

We use 4 real-world graphs, *i.e.*, *NetHEPT*, *Epinions*, *DBLP* and *LiveJournal*, which are used in previous work [13, 14, 31]. The statistics of the graphs are summarized in Table 2. *NetHEPT* [7] is an academic collaboration network and the rest three graphs are social networks, which are available in [22].

Model settings. As introduced in Section 2.1, given a graph $G = \langle V, E \rangle$, we adopt the most popular influence diffusion model, Independent Cascade (IC), for profit maximization. Following prior work [14, 35], the propagation probability $p(u,v)$ of each edge (u,v) in G is set to be the inverse of v's in-degree (*i.e.*, its number of in-neighbors). Function $f(S)$ is the expected number of users activated by S during the IC diffusion process. We adopt the *degree-proportional cost model* for cost function c [13, 34]. Specifically, the cost c_v of node v in G is proportional to its out-degree $d_{out}(v)$ (*i.e.*, its number of out-neighbors) as follows: $c_v = \lambda \cdot d_{out}(v)^\gamma$, where λ and γ are two free parameters controlling the cost model. The default settings of λ and γ are that: $\lambda = 0.2$ and $\gamma = 1$. When $d_{out}(v) = 0$, c_v is 1. The total cost of a set $S \subseteq V$ is $c(S) = \sum_{v \in S} c_v$.

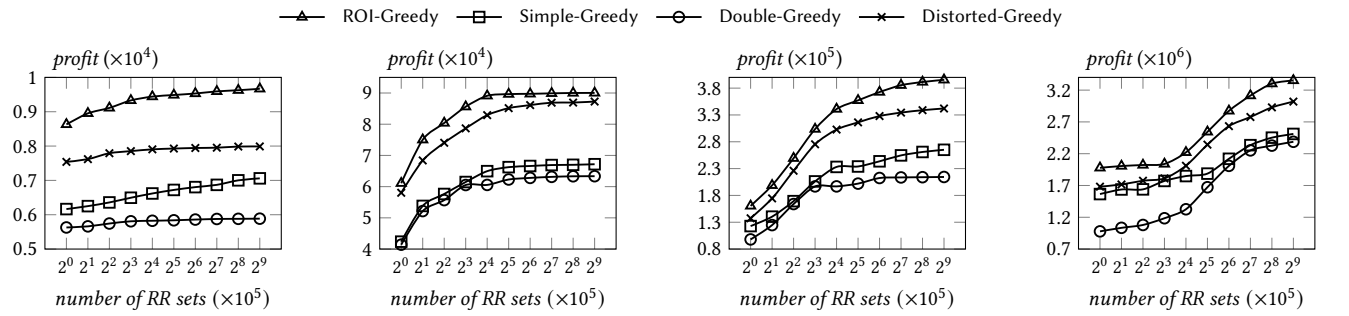
Competitors and settings. First, to evaluate the profitability, *i.e.*, $f(S) - c(S)$, we compare ROI-Greedy against three competitors including Simple-Greedy [25, 37], Double-Greedy[4] and Distorted-Greedy [12]. Specifically, we vary the number of RR sets and compare the profit of each method. $f(S)$ is obtained by 10,000 Monte Carlo simulations. We also vary γ and λ, the parameters in the degree-proportional cost model, to evaluate the robustness and profitability of all methods. We vary the number of RR sets in $\{1, 2^1, \ldots, 2^8, 2^9\} \times 10^5$, λ in $\{0.1, 0.2, \ldots, 0.5, 0.6\}$, and γ in $\{0, 0.2, \ldots, 1, 1.2\}$. The default value of the number of RR sets is $2^8 \times 10^5$. The experimental results are presented in Section 5.2.

Second, to evaluate the efficiency and effectiveness of ROI-PM for the profit maximization problem, we compare ROI-PM against an algorithm obtained by directly replacing Algorithm 3 in Algorithm 2 with Distorted-Greedy, dubbed as DGP. It is easy to prove that DGP can return results with $(1-1/e-\epsilon)f(S^*) - c(S^*)$ guarantee with high probability, by using similar proofs as Theorem 3. We set $\epsilon = 0.2$, the failure probability $\delta = 1/n$, where n is the number of nodes in the input graph, and vary λ in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ to evaluate the running time and the profit of each method. The experimental results are presented in Section 5.3. Note that Simple-Greedy, Double-Greedy, and existing solutions for profit maximization do not provide guarantees under the general problem settings studied in this paper. Therefore, they are not compared.

5.2 Evaluation of ROI-Greedy

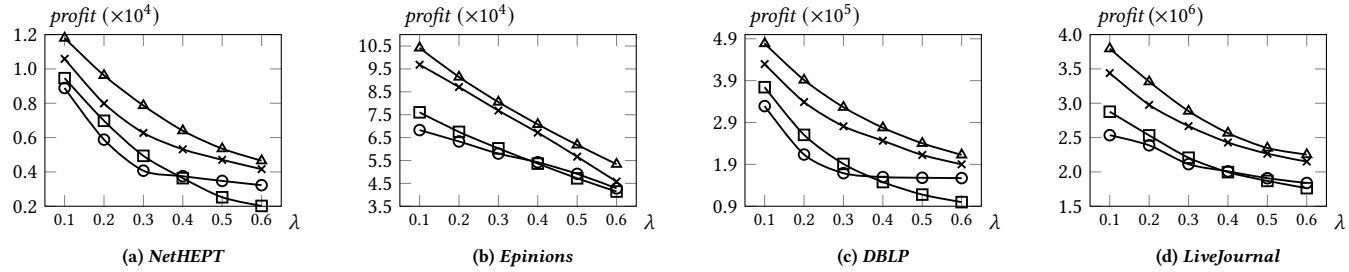
In this section, we evaluate the profitability (*i.e.*, $f(S) - c(S)$) of ROI-Greedy against three competitors including Simple-Greedy [25, 37], Double-Greedy [4] and Distorted-Greedy [12], by varying the number of RR sets, λ, and γ.

Varying the number of RR sets. Figure 1 reports the profits of all methods on the four graphs, when increasing the number of RR sets from 10^5 to $2^9 \times 10^5$. x-axis is the number of RR sets. y-axis is the profit (the higher the better). ROI-Greedy gains the highest profit under all settings over all the four graphs consistently and maintains a significant performance gap compared to all competitors for



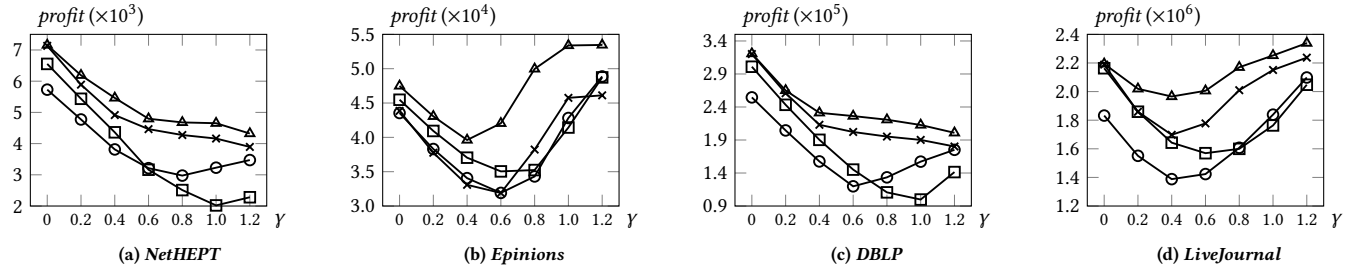
(a) ***NetHEPT*** (b) ***Epinions*** (c) ***DBLP*** (d) ***LiveJournal***

Figure 1: Profit with varying number of RR sets.



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

Figure 2: Profit with varying λ.



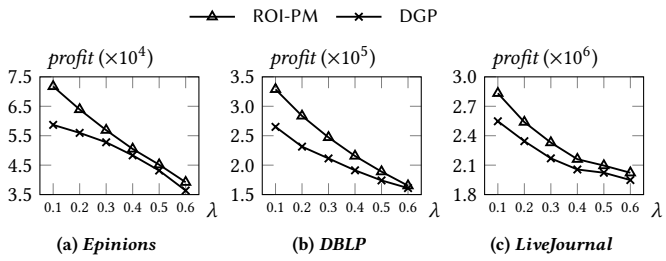
(a) ***NetHEPT*** (b) ***Epinions*** (c) ***DBLP*** (d) ***LiveJournal***

Figure 3: Profit with varying γ.

different numbers of RR sets. For instance, in Figure 1(a), compared to the best competitor Distorted-Greedy, ROI-Greedy improves the profit by up to 21% on *NetHEPT*. The profit of ROI-Greedy is also higher by up to 8.8%, 16%, and 12% than Distorted-Greedy on *Epinions*, *DBLP*, and *LiveJournal* respectively in Figures 1 (b)-(d). In Figure 1, we observe that Double-Greedy tends to underperform Simple-Greedy. The reason is that we have a small $\lambda = 0.2$ in Figure 1, indicating that the costs of nodes are small and the dominant term in $f - c$ is f. In such a case, since Simple-Greedy is good at optimizing f, Simple-Greedy is more effective than Double-Greedy. Moreover, observe that the profits of all methods increase as the number of RR sets increases, and the superiority of ROI-Greedy maintains. The experimental results are consistent with our theoretical analysis of ROI-Greedy in Section 3. In particular, instead of only focusing on return as in existing solutions, ROI-Greedy is based on a new selection criterion by considering return-on-investment (ROI), and, thus, ROI-Greedy has tighter approximation guarantee (i.e., $f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} c(S^*)$) than existing solutions, which is validated on by the experimental results on real-world graphs.

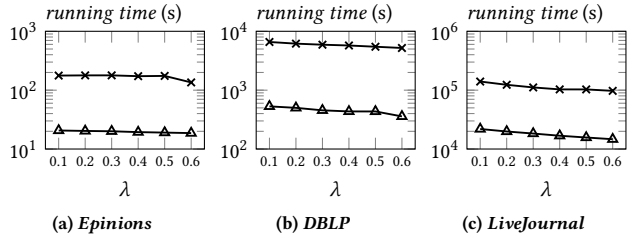
Varying λ. We vary λ while γ and the number of RR sets are set to their default values, i.e., 1 and $2^8 \times 10^5$ respectively. Figure 2 presents the profits (y-axis) of all methods on the four graphs when varying λ from 0.1 to 0.6. The overall observation is that ROI-Greedy always produces the highest profit over all graphs under all settings, compared to the competitors. When λ increases, the profits of all methods decrease since the costs of all nodes increase. The significant performance gap between ROI-Greedy and the competitors maintains. For instance, on *NetHEPT* in Figure 2(a), the profit of ROI-Greedy is higher than that of Distorted-Greedy, Simple-Greedy, and Double-Greedy by up to 25%, 130% and 93%, respectively. The experimental results demonstrate the advantage of ROI-Greedy over existing solutions in finding highly profitable solutions, by utilizing the novel return-on-investment selection metric.

Varying γ. Figure 3 reports the experimental results when varying γ in $\{0, 0.2, 0.4, 0.6, 0.8, 1, 1.2\}$, while $\lambda = 0.2$ and the number of RR sets is $2^8 \times 10^5$ as default. From Figure 3(a) to 3(d), the profit of ROI-Greedy is consistently higher than those of all competitors over all graphs under all settings. The profit of ROI-Greedy is up to 11%, 32%, 12% and 16% higher than Distorted-Greedy on *NetHEPT*, *DBLP*,



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

Figure 4: Profit with varying λ.



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

Figure 5: Running time with varying λ.

Epinions, and *LiveJournal*, respectively. For instance, on *Epinions* in Figure 3(b), when γ is 0.6, the profit of ROI-Greedy is 4.2×10^4, while the profit of Distorted-Greedy is just 3.5×10^4. All the above experiments not only demonstrate the superiority of ROI-Greedy, but also show that ROI-Greedy is stable and robust to generate the highest profit, in terms of various parameter settings.

5.3 Evaluation of ROI-PM

We evaluate the proposed ROI-PM against DGP for profit maximization on real-world graphs. In particular, we study the profit (*i.e.*, effectiveness) and running time (*i.e.*, efficiency) of these methods when varying λ in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ while setting $\epsilon = 0.2$ and $\delta = 1/n$ as default. The results when varying γ are similar and thus omitted here. We only report the results on *Epinions*, *DBLP* and *LiveJournal* due to the interest of space.

Note that most existing profit maximization solutions (e.g., [1, 13, 25, 32, 33, 36]) are based on either Simple-Greedy or Double-Greedy. However, Simple-Greedy and Double-Greedy cannot provide guarantees under the problem settings studied in this paper. Hence, these existing solutions, inheriting the drawbacks of Simple-Greedy or Double-Greedy, are not compared under the profit maximization application here. In particular, as discussed in Section 2.2, the results returned by Simple-Greedy can be arbitrarily worse than the optimal result, and the approximation guarantee of Double-Greedy relies on a strong assumption that requires for any $S' \subseteq V$, $f(S') - C(S') \geq 0$. Further, in practice, as shown in Section 5.2, Simple-Greedy and Double-Greedy have already shown inferior performance compared against the proposed ROI-Greedy. On the other hand, the ROI-Greedy-based ROI-PM does not rely on such strong assumptions and can provide worst-case approximation guarantees for profit maximization. Therefore, for the profit maximization application, we only compare ROI-PM with DGP that is based on Distorted-Greedy, which can also provide approximation guarantee under the same problem settings studied in this paper.

Profit when varying λ. Figure 4 reports the profits achieved by ROI-PM and DGP when varying λ from 0.1 to 0.6. We observe that ROI-PM consistently and significantly outperforms DGP under all settings on all datasets. The profit of ROI-PM is up to 22%, 24% and 11% higher than that of DGP on *Epinions*, *DBLP*, and *LiveJournal*, respectively, as shown in Figures 4(a,b,c). For instance, on *Epinions* in Figure 4(a), when $\lambda = 0.2$, the profit of ROI-PM is 6.4×10^4, while that of DGP is just 5.6×10^4. ROI-PM always achieves a higher profit in practice since it provides a tighter theoretical guarantee as proved in Section 4.3. When λ increases, the performance gap between ROI-PM and DGP slightly narrows. The reason is that when λ increases, the cost c_v of each node v increases, making it harder to gain profits.

Running time when varying λ. Figure 5 shows the running time of ROI-PM and DGP on *Epinions* and *DBLP* when we vary λ from 0.1 to 0.6. y-axis is the running time in seconds (s) and is in log-scale. Observe that ROI-PM is significantly faster than DGP under all settings. In particular, ROI-PM is up to 9 times faster than DGP on *Epinions*, as well as up to 14.5 and 7 times faster than DGP on *DBLP* and *LiveJournal*, respectively. For instance, when λ is 0.2, on *DBLP*, ROI-PM only needs 501 seconds (*i.e.*, 8.3 minutes) to finish, while DGP takes around 6,169 seconds (*i.e.*, 1.7 hours). Recall that the number of RR sets generated is inversely proportional to the quality of the solution constructed, for both ROI-PM and DGP. However, the efficiency gap (Figure 5) between ROI-PM and DGP is much greater than the effectiveness gap (Figure 4) between these two algorithms. This is because the early termination (Lines 5 to 8 of Algorithm 2) is more effective for ROI-PM than for DGP.

6 OTHER RELATED WORK

In addition to the methods reviewed in Section 2.2, we review other methods that are relevant to this work in the following.

As a pioneer work [28], Nemhauser *et al.* prove that the greedy algorithm ensures a $(1-1/e)$ approximate guarantee for maximizing a non-negative, monotone, and submodular function, subject to a cardinality constraint. After that, a plethora of studies are conducted to solve this problem in various settings. Specifically, given a non-negative, monotone, and submodular function f, a non-negative and modular cost function c under knapsack constraint, and a cost budget B, the problem asks for a set S that maximizes $f(S)$ subject to $c(S) \leq B$. Towards this end, a simple greedy solution is proposed in [18] and is then shown to be able to provide $(1-1/e)/2$-approximate guarantee. Besides, there also exists a large body of literature [4, 5, 9, 11, 20, 29, 30] on non-monotone submodular maximization. Among them, the two methods [11, 30] that are most related to our work focus on maximizing $g + \ell$, where g is a non-negative, monotone, and submodular function and ℓ is an arbitrary modular function. More specifically, in [30], the authors propose a randomized polynomial time algorithm that ensures $g(S) + \ell(S) \geq (1 - 1/e)g(S^*) + \ell(S^*)$, whose efficiency is subsequently improved in [11]. However, as pinpointed in [12], both algorithms [11, 30] are practically intractable and inefficient, and thus subsumed by Distorted-Greedy [12]. Additionally, several studies [12, 21, 26] aim at alleviating the efficiency issue of classic greedy methods.

As mentioned in Section 4, profit maximization is an instance of unconstrained submodular maximization. Most existing solutions for profit maximization, *e.g.*, [1, 13, 25, 32, 33, 36], adopt

Simple-Greedy and Double-Greedy methods. In [33], Tang *et al.* first point out that the results returned by Simple-Greedy can be arbitrarily worse than the optimum. Tang *et al.*[33] further show that based on the non-negativity assumption of the submodular function, the deterministic and the randomized versions of Double-Greedy ensure $\frac{1}{3}$- and $\frac{1}{2}$-approximation guarantees, respectively. However, as discussed in Section 2, this assumption is unrealistic in practice. Based on the same assumption, Huang *et al.* [13] propose an algorithm for solving the profit maximization problem in an adaptive setting, in which the set S of seed nodes is extracted from a pre-defined small set T of nodes, where $T \subset V$ and $|T| \ll |V|$.

There are many studies on sensor placement and text summarization *e.g.*, [15, 19, 21, 23, 24]. However, these studies formulated text summarization and sensor placement as constrained submodular maximization problems, which are very different from the unconstrained submodular maximization problem studied in this paper.

7 CONCLUSIONS

In this paper, we propose ROI-Greedy, a novel polynomial time algorithm for USM-MC, which returns a solution S satisfying $f(S) - c(S) \geq f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*)$, where S^* denotes the optimal solution. The worst-case guarantee of ROI-Greedy is stronger than all existing solutions. We also prove that the guarantee is tight compared with a lower bound derived in this paper. Further, we devise a non-trivial extension of ROI-Greedy to solve the profit maximization problem. Extensive experiments on benchmark datasets demonstrate that ROI-Greedy significantly outperforms competing methods in terms of efficiency and solution quality.

ACKNOWLEDGMENTS

Xiaokui Xiao is supported by the National University of Singapore under SUG grant R-252-000-686-133. Tianyuan Jin is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-PhD/2021-01-004[T]). Yu Yang is supported in part by the Hong Kong Research Grants Council (ECS 21214720) and City University of Hong Kong (Project 9610465). Jieming Shi is supported by the financial support (1-BE3T) of research project (P0033898) from Hong Kong Polytechnic University. The findings herein reflect the work, and are solely the responsibility, of the authors.

APPENDIX

Proof of Lemma 1. For convenience, we abuse the notation and let $\Delta(T \mid S) = f(T \cup S) - f(S)$ denote the marginal increase in f when we add all elements in a set $T \subseteq V$ into another set $S \subseteq V$. We have

$$\begin{aligned}\frac{\Delta(v_i \mid S_{i-1})}{c_{v_i}} &= \max_{u \in V \setminus S_{i-1}} \frac{\Delta(u \mid S_{i-1})}{c_u} \quad \geq \max_{u \in S^+ \setminus S_{i-1}} \frac{\Delta(u \mid S_{i-1})}{c_u} \\ &\geq \frac{\sum_{u \in S^+} \Delta(u \mid S_{i-1})}{\sum_{u \in S^+} c_u} \quad \geq \frac{\Delta(S^+ \mid S_{i-1})}{c(S^+)} \\ &= \frac{f(S^+ \cup S_{i-1}) - f(S_{i-1})}{c(S^+)} \geq \frac{f(S^+) - f(S_{i-1})}{c(S^+)}.\end{aligned}$$

□

Proof of Lemma 2. From Lemma 1, we have $f(S_1) = f(\{v_1\}) \geq \frac{c_{v_k} f(S^+)}{c(S^+)}$. Hence, Eq. (5) holds for $i = 1$. Now, assume that $f(S_i) \geq \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{v_k}}{c(S^+)}\right)\right) f(S^+)$ holds for any $i \leq j$. For S_{j+1}, we have

$$\begin{aligned}f(S_{j+1}) &= \Delta(v_{j+1} \mid S_j) + f(S_j) \\ &\geq \frac{c_{v_{j+1}} (f(S^+) - f(S_j))}{c(S^+)} + f(S_j) = \frac{c_{v_{j+1}}}{c(S^+)} f(S^+) + \left(1 - \frac{c_{v_{j+1}}}{c(S^+)}\right) f(S_j) \\ &\geq \frac{c_{v_{j+1}}}{c(S^+)} f(S^+) + \left(1 - \frac{c_{v_{j+1}}}{c(S^+)}\right) \cdot \left(1 - \prod_{k=1}^{j} \left(1 - \frac{c_{v_k}}{c(S^+)}\right)\right) f(S^+) \\ &= \left(1 - \prod_{k=1}^{j+1} \left(1 - \frac{c_{v_k}}{c(S^+)}\right)\right) f(S^+).\end{aligned}$$

By induction, Eq. (5) holds. □

Proof of Lemma 3. We consider two cases.

First, if $\beta \leq c(S^+)$, we first prove for any $k \leq t-1$, $c_{v_k} \leq c(S^+)$. Suppose for contradiction that there exists $k \leq t-1$ such that $c_{v_k} \geq c(S^+)$, then

$$\begin{aligned}f(S_k) &= \Delta(v_k \mid S_{k-1}) + f(S_{k-1}) \\ &\geq \frac{c_{v_k}}{c(S^+)} (f(S^+) - f(S_{k-1})) + f(S_{k-1}) \geq f(S^+),\end{aligned} \tag{19}$$

where the first inequality is due to Lemma 1 and the second inequality is due to Eq.(8). Therefore, $f(S_{t-1}) \geq f(S_k) \geq f(S^+)$, which contradicts with Eq.(8). Hence, for all $k \leq t-1$, $c_{v_k} \leq c(S^+)$. Now, we have

$$\begin{aligned}&\frac{\beta(f(S^+) - f(S_{t-1}))}{c(S^+)} + f(S_{t-1}) \\ &\geq \frac{\beta}{c(S^+)} f(S^+) + \left(1 - \frac{\beta}{c(S^+)}\right) \left(1 - \prod_{k=1}^{t-1} \left(1 - \frac{c_{v_k}}{c(S^+)}\right)\right) f(S^+) \\ &= \left(1 - \left(1 - \frac{\beta}{c(S^+)}\right) \prod_{k=1}^{t-1} (1 - \frac{c_{v_k}}{c(S^+)})\right) f(S^+) \\ &\geq \left(1 - \left(1 - \frac{\beta + \sum_{k=1}^{t-1} c_{v_k}}{t \cdot c(S^+)}\right)^t\right) f(S^+) \\ &= \left(1 - \left(1 - \frac{\ln(f(S^+)/c(S^+))}{t}\right)^t\right) f(S^+) \\ &\geq f(S^+) - c(S^+),\end{aligned} \tag{20}$$

where the first inequality is by Lemma 2 and $\beta \leq c(S^+)$, and the second inequality is due to Eq. (6), $c_{v_k} \leq c(S^+)$ and $\beta + \sum_{k=1}^{t-1} c_{v_k} = \ln \frac{f(S^+)}{c(S^+)} \cdot c(S^+)$, while the third inequality holds because

$$\left(1 - \frac{\ln\left(f(S^+)/c(S^+)\right)}{t}\right)^t \leq e^{-\ln \frac{f(S^+)}{c(S^+)}} = \frac{c(S^+)}{f(S^+)}.$$

Second, if $\beta > c(S^+)$, then

$$\begin{aligned}&\frac{\beta(f(S^+) - f(S_{t-1}))}{c(S^+)} + f(S_{t-1}) \\ &= f(S^+) + \left(\frac{\beta}{c(S^+)} - 1\right) (f(S^+) - f(S_{t-1})) \\ &\geq f(S^+) \geq f(S^+) - c(S^+),\end{aligned} \tag{21}$$

since $\beta > c(S^+)$ and $f(S_{t-1}) < f(S^+)$. □

Proof of Lemma 6. Let $\mathcal{E}_{1i}$ denote the event that Eq. (16) holds for the i-th iteration, and $\mathcal{E}_{2i}$ denote the event that Eq. (17) holds for the i-th iteration. Let ϵ' be the solution to the equation

$$\exp\left(-\frac{(\epsilon')^2}{2+\epsilon'} \frac{\theta_i}{n} f(S^\circ)\right) = \frac{\delta}{6i^2}. \tag{22}$$

Let $\mathcal{H}_1$ be the event

$$f_{\mathcal{R}_2}(S^\circ) \le (1+\epsilon')f(S^\circ). \tag{23}$$

Then, by Lemma 4, $\mathbb{P}[\mathcal{H}_1] \ge 1-\delta/(6i^2)$. When $\mathcal{H}_1$ occurs, by Eq. (22),

$$\frac{(\epsilon')^2}{(2+\epsilon')(1+\epsilon')} = \frac{n\ln(6i^2/\delta)}{\theta_i\cdot(1+\epsilon')\cdot f(S^\circ)} \le \frac{n\ln(6i^2/\delta)}{\theta_i\cdot f_{\mathcal{R}_2}(S^\circ)}. \tag{24}$$

By the definition of ϵ_1, we have $\epsilon_1 \ge \epsilon'$ given $\mathcal{H}_1$. Hence,

$$\mathcal{H}_1 \Rightarrow f_{\mathcal{R}_2}(S^\circ) \le (1+\epsilon_1)f(S^\circ).$$

This leads to

$$\mathbb{P}\big[f_{\mathcal{R}_2}(S^\circ) \le (1+\epsilon_1)f(S^\circ)\big] \ge \mathbb{P}[\mathcal{H}_1] \ge 1-\delta/(6i^2).$$

Therefore, $\mathbb{P}[\mathcal{E}_{1i}] \ge 1-\delta/(6i^2)$.

Similarly, for $f(S^*)$, let $\bar{\epsilon}$ be the solution to the equation

$$\exp\left(-\frac{\bar{\epsilon}^2}{2}\frac{\theta_i}{n}f(S^*)\right) = \frac{\delta}{6i^2},$$

and $\mathcal{H}_2$ be the event $f_{\mathcal{R}_1}(S^*) \le (1-\bar{\epsilon})f(S^*)$. From Lemma 4, we have $\mathbb{P}[\mathcal{H}_2] \ge 1-\delta/(6i^2)$. Conditioned on $\mathcal{E}_{1i}$, we have

$$\begin{aligned}\frac{\bar{\epsilon}^2}{2(1+\epsilon_1)} &= \frac{n}{\theta_i\cdot(1+\epsilon_1)\cdot f(S^*)}\ln\left(\frac{6i^2}{\delta}\right)\\ &\le \frac{n}{\theta_i\cdot(1+\epsilon_1)\cdot(f(S^\circ)-c(S^\circ))}\ln\left(\frac{6i^2}{\delta}\right)\\ &\le \frac{n}{\theta_i\cdot(f_{\mathcal{R}_2}(S^\circ)-(1+\epsilon_1)c(S^\circ))}\ln\left(\frac{6i^2}{\delta}\right),\end{aligned}$$

where the first inequality is due to the definition of $\bar{\epsilon}$, and the last inequality is due to $\mathcal{E}_{1i}$. From the definition of ϵ_2, conditioned on $\mathcal{E}_{1i}$, we have $\epsilon_2 \ge \bar{\epsilon}$. Finally, we have

$$\mathbb{P}\big[f_{\mathcal{R}_1}(S^*) \ge (1-\epsilon_2)f(S^*) \mid \mathcal{E}_{1i}\big] \ge \mathbb{P}[H_2] \ge 1-\frac{\delta}{6i^2}.$$

Thus, $\mathbb{P}[\mathcal{E}_{2i} \mid \mathcal{E}_{1i}] \ge 1-\delta/(6i^2)$. As a result,

$$\mathbb{P}[\mathcal{E}_{2i}\cap\mathcal{E}_{1i}] = \mathbb{P}[\mathcal{E}_{2i}\mid\mathcal{E}_{1i}]\mathbb{P}[\mathcal{E}_{1i}] \ge 1-\frac{\delta}{3i^2}.$$

For all iterations, we have

$$\begin{aligned}\mathbb{P}\left[\bigcap_{i=1}^{\infty}\mathcal{E}_{1i}\bigcap_{i=1}^{\infty}\mathcal{E}_{2i}\right] &\ge \prod_{i=1}^{\infty}\mathbb{P}[\mathcal{E}_{2i}\cap\mathcal{E}_{1i}] \ge \prod_{i=1}^{\infty}\left(1-\frac{\delta}{3i^2}\right)\\ &\ge 1-\sum_{i=1}^{\infty}\frac{\delta}{3i^2} \ge 1-\frac{\pi^2\cdot\delta}{18} \ge 1-\frac{5\delta}{9},\end{aligned} \tag{25}$$

where the third inequality follows from the Weierstrass product inequality. □

Proof of Theorem 3. We consider two cases depending on whether Line 9 in Algorithm 2 is triggered.

Case 1: Line 9 is triggered. Then, in the last iteration, we have

$$(t-1)/t + \epsilon_2 + \epsilon_1 \le \epsilon, \text{ and} \tag{26}$$

$$\epsilon_1 + \epsilon_2 \le \epsilon, \tag{27}$$

where $\epsilon_1, \epsilon_2 \in (0,1)$ and $t > 0$. Suppose that both Eq. (16) and (17) hold. Then,

$$\begin{aligned}f_{\mathcal{R}_1}(S^\circ) - c(S^\circ) &\ge f_{\mathcal{R}_1}(S^*) - c(S^*) - \ln\tfrac{f_{\mathcal{R}_1}(S^*)}{c(S^*)}\cdot c(S^*)\\ &\ge (1-\epsilon_2)f(S^*) - c(S^*) - \ln\tfrac{f(S^*)}{c(S^*)}\cdot c(S^*),\end{aligned} \tag{28}$$

where the first inequality is due to Theorem 1, and the second inequality is due to Eq. (17) and Lemma 5. Moreover, by Eq.(16),

$$\begin{aligned}f_{\mathcal{R}_1}(S^\circ) - c(S^\circ) &= t(f_{\mathcal{R}_2}(S^\circ) - c(S^\circ))\\ &\le t(f(S^\circ) - c(S^\circ)) + t\epsilon_1 f(S^\circ).\end{aligned} \tag{29}$$

Finally, we have

$$f(S^\circ) - c(S^\circ) \ge 1/t\left(f_{\mathcal{R}_1}(S^\circ) - c(S^\circ)\right) - \epsilon_1 f(S^*) \tag{30}$$

$$\ge 1/t\left((1-\epsilon_2)f(S^*) - c(S^*) - \ln\frac{f(S^*)}{c(S^*)}\cdot c(S^*)\right) - \epsilon_1 f(S^*) \tag{31}$$

where the first inequality is from Eq. (29), and second inequality is from Eq. (28).

If $1/t < 1$, following Eq. (31), we have

$$\begin{aligned}&f(S^\circ) - c(S^\circ)\\ &\ge 1/t\left((1-\epsilon_2)f(S^*) - c(S^*) - \ln\frac{f(S^*)}{c(S^*)}\cdot c(S^*)\right) - \epsilon_1 f(S^*)\\ &\ge f(S^*) - c(S^*) + (1/t-1)f(S^*) - (\epsilon_2+\epsilon_1)f(S^*) - \ln\frac{f(S^*)}{c(S^*)}c(S^*)\\ &\ge (1-\epsilon)f(S^*) - c(S^*) - \ln\frac{f(S^*)}{c(S^*)}\cdot c(S^*),\end{aligned}$$

where the last inequality is from Eq. (26).

Otherwise, $1/t \ge 1$, following Eq. (30), we have

$$\begin{aligned}f(S^\circ) - c(S^\circ) &\ge f_{\mathcal{R}_1}(S^\circ) - c(S^\circ) - \epsilon_1 f(S^*)\\ &\ge (1-\epsilon_1-\epsilon_2)f(S^*) - c(S^*) - \ln\frac{f(S^*)}{c(S^*)}\cdot c(S^*)\\ &\ge (1-\epsilon)f(S^*) - c(S^*) - \ln\frac{f(S^*)}{c(S^*)}\cdot c(S^*),\end{aligned}$$

where the second inequality is from Eq. (28), and the last inequality is from Eq.(27).

By Lemma 6, both Eq. (16) and (17) hold with probability at least $1-\frac{5\delta}{9}$. Thus, when Line 9 is triggered, with probability at least $1-\frac{5\delta}{9}$, Eq. (18) holds.

Case 2: Line 9 is not triggered. In this case, we have

$$\theta_i = |\mathcal{R}_1| > (8+2\epsilon)(1+\epsilon_1)n\frac{\ln(6/\delta)+n\ln 2}{\epsilon^2\max\{1, f_{\mathcal{R}_2}(S^\circ)-(1+\epsilon_1)c(S^\circ)\}}$$

when Algorithm 2 terminates.

Notice that when the event $\cap_i\mathcal{E}_{1i}$ occurs. $\cap_i\mathcal{E}_{1i}$ implies that

$$\begin{aligned}\max\{1, f_{\mathcal{R}_2}(S^\circ)-(1+\epsilon_1)c(S^\circ)\} &\le \max\{1, (1+\epsilon_1)(f(S^\circ)-c(S^\circ))\}\\ &\le (1+\epsilon_1)f(S^*).\end{aligned}$$

Then, we have

$$\theta_i = |\mathcal{R}_1| \ge (8+2\epsilon)n\frac{\ln(6/\delta)+n\ln 2}{\epsilon^2 f(S^*)}$$

when Algorithm 2 terminates. Let $x = \epsilon f(S^*)/(2f(S))$. In this case, by Lemma 4, for any $S \subseteq V$,

$$\begin{aligned}\mathbb{P}[f_{\mathcal{R}_1}(S) - f(S) \ge \frac{\epsilon}{2}f(S^*)] &\le \exp\left(-\frac{x^2}{2+x}\frac{\theta_i}{n}f(S)\right)\\ &\le \exp\left(-\frac{\epsilon^2}{8+2\epsilon}\frac{\theta_i}{n}f(S^*)\right) \le \frac{\delta}{6\cdot 2^n},\end{aligned}$$

where the second inequality is due to the fact that the right side of the first inequality achieves its maximum at $f(S) = f(S^*)$. Similarly, we also have $\mathbb{P}[f_{\mathcal{R}_1}(S) - f(S) \le -\frac{\epsilon}{2}f(S^*)] \le \frac{\delta}{6\cdot 2^n}$. Since $S \in 2^V$ and $|2^V| = 2^n$, $\mathbb{P}[|f_{\mathcal{R}_1}(S) - f(S)| \le \frac{\epsilon}{2}f(S^*), \forall S \subseteq V] \ge 1-\frac{\delta}{3}$. When $|f_{\mathcal{R}_1}(S) - f(S)| \le \frac{\epsilon}{2}f(S^*)$ for all $S \subseteq V$, we have

$$f_{\mathcal{R}_1}(S^*) \ge (1-\frac{\epsilon}{2})f(S^*), \text{ and} \tag{32}$$

$$f_{\mathcal{R}_1}(S^\circ) \le f(S^\circ) + \frac{\epsilon}{2}f(S^*). \tag{33}$$

Thus, when the event $\cap_i \mathcal{E}_{1i}$ happens, we have

$$\begin{aligned} f(S^\circ) - c(S^\circ) &\geq f_{\mathcal{R}_1}(S^\circ) - c(S^\circ) - \frac{\epsilon}{2} f(S^*) \\ &\geq f_{\mathcal{R}_1}(S^*) - c(S^*) - \ln \frac{f_{\mathcal{R}_1}(S^*)}{c(S^*)} \cdot c(S^*) - \frac{\epsilon}{2} f(S^*) \\ &\geq (1-\epsilon) f(S^*) - c(S^*) - \ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*) \end{aligned}$$

where the third inequality is due to Eq. (32) and Lemma 5.

According to Eq. (25), the event $\cap_i \mathcal{E}_{1i}$ occurs with probability at least $1 - \frac{5\delta}{9}$. Therefore, when Line 9 is not triggered, with probability at least $1 - \frac{5\delta}{9} - \frac{\delta}{3} \geq 1 - \delta$, we have

$$f(S^\circ) - c(S^\circ) \geq (1-\epsilon) f(S^*) - c(S^*) - \frac{f(S^*)}{c(S^*)} \cdot c(S^*).$$

Combining **Case 1** and **Case 2**, the theorem is proved. □

Proof of Theorem 2. We use again the reduction from *minimum set cover* demonstrated in Section 3.1. Assume that there is an algorithm Alg satisfying Eq. (13). We show that we can use algorithm Alg to achieve an approximate ratio of $c \ln N$ ($N = |\cup_{i=1}^n u_i|$) for minimum set cover for some $c < 1$, which contradicts the assumption that P≠NP [8].

First, on an arbitrary minimum set cover instance $V = \{u_1, ..., u_n\}$, recall that S^*, the optimal solution to USM-MC, is a minimum set cover. Thus, we have $f(S^*) \geq 2c(S^*)$ and $c(S^*) \ln \frac{f(S^*)}{c(S^*)} \leq \frac{f(S^*)}{e}$. Let $\bar{\epsilon} = \left(\frac{1}{2} - \frac{1}{e}\right)\epsilon$. We have $\bar{\epsilon} > 0$ since $\epsilon > 0$. Then, by Eq. (13),

$$\begin{aligned} & f(S') - c(S') \\ &\geq \epsilon \cdot \left(f(S^*) - c(S^*) - \ln \tfrac{f(S^*)}{c(S^*)} \cdot c(S^*) \right) \\ &\quad + \left(f(S^*) - c(S^*) - \ln \tfrac{f(S^*)}{c(S^*)} \cdot c(S^*) \right) \\ &\geq (\tfrac{1}{2} - \tfrac{1}{e})\epsilon \cdot f(S^*) + \left(f(S^*) - c(S^*) - \ln \tfrac{f(S^*)}{c(S^*)} \cdot c(S^*) \right) \\ &= (1+\bar{\epsilon}) f(S^*) - c(S^*) - \ln \tfrac{f(S^*)}{c(S^*)} \cdot c(S^*). \end{aligned} \tag{34}$$

Consider the USM-MC problem on V, with $f(S) = |\cup_{u_i \in S} u_i|$ for all $S \subseteq V$, and $c_{u_i} = \frac{1}{2}$ for all $i = 1, \ldots, n$. Suppose that we use Alg to solve this USM-MC problem, and obtain a solution $S^1 \subseteq V$. We consider two cases depending on whether S^1 is a set cover.

Case 1: S^1 is already a set cover of V. Recall that in Section 3.1, we prove that the minimum set cover S^* of V is an optimal solution to USM-MC. By Eq. (34),

$$\begin{aligned} f(S^1) - c(S^1) &\geq f(S^*) - c(S^*) - \ln \tfrac{f(S^*)}{c(S^*)} \cdot c(S^*) + \bar{\epsilon} \cdot f(S^*) \\ &\geq f(S^*) - c(S^*) - (1 - e\bar{\epsilon}) \cdot \ln \tfrac{f(S^*)}{c(S^*)} \cdot c(S^*), \end{aligned}$$

where the last inequality is due to $f(S^*)/c(S^*) \geq 2$ and $\ln \frac{f(S^*)}{c(S^*)} \cdot c(S^*) \leq \frac{f(S^*)}{e}$. Since both S^1 and S^* are set covers, we have $f(S^1) = f(S^*) = |\cup_{i=1}^n u_i| = N$. Moreover, we have $c(S^1) = \frac{1}{2}|S^1|$ and $c(S^*) = \frac{1}{2}|S^*|$. Therefore, $|S^1| \leq |S^*|\big((1 - e\bar{\epsilon})(\ln N - \ln |S^*| + \ln 2) + 1\big) \leq |S^*| \left((1 - e\bar{\epsilon}) \ln N + O(1)\right)$. This indicates that there exists some constant $\epsilon' > 0$ such that

$$|S^1| \leq |S^*|(1 - \epsilon') \ln N. \tag{35}$$

Case 2: S^1 is not a set cover of V. Let $b_1 = \cup_{u_i \in S^1} u_i$. For convenience, we rename $V = \{u_1, u_2, \ldots, u_n\}$ to $V^1 = \{u_1^1, u_2^1, \ldots, u_n^1\}$. We construct another set cover instance $V^2 = \{u_1^2, u_2^2, \ldots, u_n^2\}$ where $u_i^2 = u_i^1 \setminus b_1$. Let $f^2(S) = |\cup_{u_i^2 \in S} u_i^2|$. Suppose that we apply Alg on V^2 with $f^2(\cdot)$, and obtain a solution S^2. If S^2 is still not a set cover of V^2, we can repeat the same process to construct new set cover instances $V^3, \ldots, V^l$ with $f^3(\cdot), \ldots, f^l(\cdot)$, and to apply Alg to derive $S^3, \ldots, S^l$ until S^l is a set cover of V^l.

Let $S^\diamond = \{u_i \mid \exists j, u_i^j \in S^j\} \subseteq V$. It can be verified that $S^\diamond$ is a set cover of V, the original set cover instance. Moreover, constructing $S^\diamond$ takes polynomial time, since Alg is a polynomial time algorithm and $l \leq N$.

Let S_j^* be the minimum set cover of V^j, and $z^j = |S^j|/|S_j^*|$. Denote $y^j = |\cup_{u_i^j \in S^j} u_i^j| / |\cup_{i=1}^n u_i^j| = f^j(S^j)/f^j(S_j^*)$ as the fraction of items covered by S^j in V^j. We prove that for $j = 1, 2, \ldots, l-1$

$$y^j \geq 1 - e^{-z^j} + \bar{\epsilon}. \tag{36}$$

Assume to the contrary that $y^j < 1 - e^{-z^j} + \bar{\epsilon}$. Note that $c(S^j) = z^j c(S_j^*)$. We have

$$\begin{aligned} f^j(S^j) - c(S^j) &< (1 - e^{-z^j} + \bar{\epsilon}) f^j(S_j^*) - z^j c(S_j^*) \\ &\leq \max_{x \in \mathbb{R}^+} \left((1 - e^{-x} + \bar{\epsilon}) f^j(S_j^*) - x c(S_j^*) \right) \\ &= (1 + \bar{\epsilon}) f^j(S_j^*) - c(S_j^*) - \ln \tfrac{f^j(S_j^*)}{c(S_j^*)} \cdot c(S_j^*), \end{aligned}$$

which contradicts Eq. (34). Therefore, Eq. (36) holds for all $j = 1, 2, \ldots, l-1$ and

$$z^j \leq \ln \frac{1}{1 - y^j + \bar{\epsilon}} < (1 - \bar{\epsilon}) \ln \frac{1}{1 - y^j}, \tag{37}$$

where the last inequality holds because $\ln \frac{1}{a + \bar{\epsilon}} < (1 - \bar{\epsilon}) \ln \frac{1}{a}$ for any $0 < a < 1$ and $0 < \bar{\epsilon} < 1$.

Let w be the total number of items in the sets in the last set cover instance V^l. Note that S^l is a set cover of V^l. By the same analysis as in **Case 1**, we have

$$|S^l| \leq |S_l^*|(1 - \epsilon') \ln w, \tag{38}$$

for some $\epsilon' > 0$. At the beginning, we have N items in the sets of V^1. In each round $j = 2, 3, \ldots, l$, we remove a fraction y^{j-1} of items from the set cover instance of the $(j-1)$-th round. Thus, we have

$$N \prod_{j=1}^{l-1} (1 - y^j) = w. \tag{39}$$

Observe that S_j^* is still a set cover of V^{j+1}, but its size may not be minimum. Thus, we have $|S_j^*| \leq |S_1^*| = |S^*|$ for all $j = 1, 2, \ldots, l$. Combining Eq. (37), (38), and (39), we have

$$\begin{aligned} \sum_{j=1}^{l} |S^j| &\leq \sum_{j=1}^{l-1} z^j |S_j^*| + (1 - \epsilon') |S_l^*| \ln w \\ &\leq (1 - \bar{\epsilon}) \sum_{j=1}^{l-1} \ln \frac{1}{1 - y^j} |S_j^*| + (1 - \epsilon') |S_l^*| \ln w \\ &\leq (1 - \min\{\bar{\epsilon}, \epsilon'\}) |S^*| \left(\sum_{j=1}^{l-1} \ln \frac{1}{1 - y^j} + \ln w \right) \\ &= |S^*| (1 - \min\{\bar{\epsilon}, \epsilon'\}) \ln N. \end{aligned} \tag{40}$$

Therefore, for the set cover $S^\diamond$ constructed from $S^1, \ldots, S^j$, we have $|S^\diamond| \leq \sum_{j=1}^{l} |S^j| \leq |S^*|(1 - \min\{\bar{\epsilon}, \epsilon'\}) \ln N$. Combining the results from **Case 1** and **Case 2**, we prove that Alg can be used to provide a $c \ln N$ approximation for minimum set cover for some $c < 1$, which leads to a contradiction. □

REFERENCES

[1] David Arthur, Rajeev Motwani, Aneesh Sharma, and Ying Xu. 2009. Pricing strategies for viral marketing on social networks. In *WINE*. 101–112.

[2] Song Bian, Qintian Guo, Sibo Wang, and Jeffrey Xu Yu. 2020. Efficient algorithms for budgeted influence maximization on massive social networks. *PVLDB* 13, 9 (2020), 1498–1510.

[3] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *SODA*. 946–957.

[4] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. 2012. A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization. In *FOCS*. 649–658.

[5] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. 2014. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.* 43, 6 (2014), 1831–1879.

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

[7] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *SIGKDD*. 199–208.

[8] Irit Dinur and David Steurer. 2014. Analytical approach to parallel repetition. In *STOC*. 624–633.

[9] Alina Ene and Huy L Nguyen. 2016. Constrained submodular maximization: Beyond 1/e. In *FOCS*. 248–257.

[10] Uriel Feige. 1998. A threshold of ln n for approximating set cover. *JACM* 45, 4 (1998), 634–652.

[11] Moran Feldman. 2020. Guess free maximization of submodular and linear sums. *Algorithmica* (2020), 1–26.

[12] Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. 2019. Submodular Maximization beyond Non-negativity: Guarantees, Fast Algorithms, and Applications. In *ICML*. 2634–2643.

[13] Keke Huang, Jing Tang, Xiaokui Xiao, Aixin Sun, and Andrew Lim. 2020. Efficient approximation algorithms for adaptive target profit maximization. In *ICDE*. 649–660.

[14] Keke Huang, Sibo Wang, Glenn Bevilacqua, Xiaokui Xiao, and Laks VS Lakshmanan. 2017. Revisiting the stop-and-stare algorithms for influence maximization. *PVLDB* 10, 9 (2017), 913–924.

[15] Rishabh Iyer and Jeff Bilmes. 2013. Submodular optimization with submodular cover and submodular Knapsack constraints. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*. 2436–2444.

[16] Ehsan Kazemi, Shervin Minaee, Moran Feldman, and Amin Karbasi. 2020. Regularized Submodular Maximization at Scale. *arXiv preprint arXiv:2002.03503* (2020).

[17] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*. 137–146.

[18] Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Information processing letters* 70, 1 (1999), 39–45.

[19] Andreas Krause and Carlos Guestrin. 2007. Near-optimal observation selection using submodular functions. In *AAAI*. 1650–1654.

[20] Jon Lee, Maxim Sviridenko, and Jan Vondrák. 2009. Submodular maximization over multiple matroids via generalized exchange properties. In *APPROX-RANDOM*. 244–257.

[21] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *SIGKDD*. 420–429.

[22] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[23] Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *ACL*. 912–920.

[24] Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *ACL*. 510–520.

[25] Wei Lu and Laks VS Lakshmanan. 2012. Profit maximization over social networks. In *ICDM*. 479–488.

[26] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. 2015. Lazier than lazy greedy. In *AAAI*. 1812–1818.

[27] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press.

[28] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.

[29] Maxim Sviridenko. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters* 32, 1 (2004), 41–43.

[30] Maxim Sviridenko, Jan Vondrák, and Justin Ward. 2017. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Mathematics of Operations Research* 42, 4 (2017), 1197–1218.

[31] Jing Tang, Keke Huang, Xiaokui Xiao, Laks VS Lakshmanan, Xueyan Tang, Aixin Sun, and Andrew Lim. 2019. Efficient approximation algorithms for adaptive seed minimization. In *SIGMOD*. 1096–1113.

[32] Jing Tang, Xueyan Tang, and Junsong Yuan. 2016. Profit maximization for viral marketing in online social networks. In *ICNP*. IEEE, 1–10.

[33] Jing Tang, Xueyan Tang, and Junsong Yuan. 2017. Profit maximization for viral marketing in online social networks: Algorithms and analysis. *TKDE* 30, 6 (2017), 1095–1108.

[34] Jing Tang, Xueyan Tang, and Junsong Yuan. 2018. Towards profit maximization for online social network providers. In *INFOCOM*. 1178–1186.

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

[36] Yuqing Zhu, Deying Li, Ruidong Yan, Weili Wu, and Yuanjun Bi. 2017. Maximizing the influence and profit in social networks. *IEEE Transactions on Computational Social Systems* 4, 3 (2017), 54–64.

[37] Yuqing Zhu, Zaixin Lu, Yuanjun Bi, Weili Wu, Yiwei Jiang, and Deying Li. 2013. Influence and profit: Two sides of the coin. In *ICDM*. 1301–1306.