Improved Evolutionary Algorithms for Submodular Maximization with Cost Constraints *

Yanhui Zhu , **Samik Basu** , **A. Pavan**
Department of Computer Science, Iowa State University, Ames, IA, USA
yanhui@iastate.edu, {sbasu, pavan}@cs.iastate.edu

Abstract

We present an evolutionary algorithm EVO-SMC for the problem of Submodular Maximization under Cost constraints (SMC). Our algorithm achieves $1/2$-approximation with a high probability $1 - 1/n$ within $\mathcal{O}(n^2 K_\beta)$ iterations, where K_β denotes the maximum size of a feasible solution set with cost constraint β. To the best of our knowledge, this is the best approximation guarantee offered by evolutionary algorithms for this problem. We further refine EVO-SMC, and develop ST-EVO-SMC. This stochastic version yields a significantly faster algorithm while maintaining the approximation ratio of $1/2$, with probability $1 - \epsilon$. The required number of iterations reduces to $\mathcal{O}(nK_\beta \log(1/\epsilon)/p)$, where the user defined parameters $p \in (0, 1]$ represents the stochasticity probability, and $\epsilon \in (0, 1]$ denotes the error threshold. Finally, the empirical evaluations carried out through extensive experimentation substantiate the efficiency and effectiveness of our proposed algorithms. Our algorithms consistently outperform existing methods, producing higher-quality solutions.

1 Introduction

A function f defined over a ground set V is submodular if, for any subsets S and T of V where $S \subseteq T$, for any $x \in V \setminus T$, $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$. The function f is monotone if $f(T) \geq f(S)$ when S is a subset of T. Monotone submodular function optimization is a fundamental problem in combinatorial optimization, applied extensively across diverse fields such as feature compression, deep learning, sensor placement, and information diffusion, among others [Bateni *et al.*, 2019; El Halabi *et al.*, 2022; Li *et al.*, 2023; Kempe *et al.*, 2003; Zhu *et al.*, 2024]. Over the last decade, various versions of submodular optimization problems have garnered substantial attention.

A typical monotone submodular maximization problem involves finding a set $S \subset V$ such that $f(S)$ is maximized, subject to a constraint that the set S belongs to a family of sets $\mathcal{I}$ known as a feasible family. An example of $\mathcal{I}$ is the set of all sets of size at most k, known as the cardinality constraint. Although many constrained submodular maximization problems are known to be NP-hard, several variants admit efficient approximation algorithms and have been amenable to rigorous theoretical analysis. This work specifically focuses on monotone submodular maximization under cost/knapsack constraints (SMC). Here, in addition to the monotone submodular function f, there is a modular cost function c over the ground set V. The cost function has the property that $c(S) = \sum_{x \in S} c(x)$. The objective of SMC is to identify a set S maximizing $f(S)$ while ensuring that $c(S)$ remains within a prescribed budget β. When the cost function is uniform, SMC reduces to the classical cardinality constraint submodular maximization problem. The works of [Khuller *et al.*, 1999; Krause and Guestrin, 2005] proposed greedy algorithms for SMC that achieve an approximation ration of $\frac{1}{2}(1 - 1/e)$, using $\mathcal{O}(n^2)$ calls to the underlying submodular function f. The work of [Sviridenko, 2004] provided an algorithm that achieves a tight $(1 - 1/e)$ approximation ratio for the SMC problem but has a very high time complexity of $\mathcal{O}(n^5)$. Subsequent research efforts have introduced variants of the greedy algorithm aiming to improve runtime at a slight expense of the approximation quality [Feldman *et al.*, 2022; Yaroslavtsev *et al.*, 2020; Li *et al.*, 2022; Ene and Nguyen, 2019; Badanidiyuru and Vondrák, 2014; Tang *et al.*, 2020].

Greedy algorithms construct solution sets iteratively by "greedily" adding one element during each iteration. This process continues as long as the underlying constraint is satisfied. While these algorithms can be rigorously analyzed, providing approximation guarantees for solution quality, they do have drawbacks. Greedy algorithms can get stuck in a local optimum. Moreover, greedy algorithms are fixed-time algorithms – meaning they can only be executed for a fixed number of iterations; even in scenarios where more computational resources (time) can be afforded, they will not produce higher-quality solutions.

Another approach to submodular optimization involves evolutionary algorithms. These algorithms mimic the population evolution procedure involving random *mutations*. These algorithms behave as follows. They maintain a set of feasible candidate solution sets. During each iteration, a candidate set is randomly selected for mutation. If the mutated set yields

*An extended version of this paper can be found: `https://arxiv.org/pdf/2405.05942` a higher-quality solution set (satisfying the constraint), then it replaces the lower-quality solution set. This approach is appealing because the random mutations can aid in moving away from the local optima. Moreover, there is no apriori bound on the number of iterations. If resources permit, the algorithm can be run for a much longer time and potentially could produce higher-quality solutions.

The work of [Qian *et al.*, 2017] used a evolutionary algorithm framework Pareto Optimization to achieve a $\frac{1}{2}(1 - 1/e)$ approximate solution to SMC. However, to guarantee this approximation ratio, the algorithm has to run exponentially many iterations. A subsequent work [Bian *et al.*, 2020] proposed an evolutionary algorithm EAMC with improved runtime. They proved that when the algorithm EAMC is run for $\mathcal{O}(n^2 K_\beta)$ iterations then the produced solution has an approximation ratio of $\frac{1}{2}(1 - 1/e)$. Furthermore, empirical results of their work showed that EAMC produces solutions significantly better than those produced by greedy-based approaches. The approximation ratio of $\frac{1}{2}(1 - 1/e)$ that is achieved by these evolutionary algorithms is not competitive compared with the best known $(1 - 1/e)$ approximation ratio by the greedy algorithm (though with a much higher time complexity). Therefore, a significant objective is to design novel and efficient evolutionary algorithms that offer stronger approximation ratios.

1.1 Our Contributions

We design an evolutionary algorithm EVO-SMC that achieves an approximation ratio of $1/2$. This marks a significant enhancement in the approximation guarantees for evolutionary algorithms applied to SMC. The $1/2$ approximation ratio is attained when the algorithm is executed for $\mathcal{O}(n^2 K_\beta)$ iterations, which is still cubic. To address this, we refine EVO-SMC and develop ST-EVO-SMC, which has a $1/2$ approximation with probability $1 - \epsilon$ and requires only $\mathcal{O}(nK_\beta \ln(1/\epsilon)/p)$ iterations. In this algorithm, $p \in (0, 1]$ is the stochasticity probability that controls the candidate set selection. When $p = 0$, ST-EVO-SMC reduces to the original EVO-SMC. Algorithm ST-EVO-SMC improves the running time of EVO-SMC by a magnitude of n. Table 1 compares the approximation ratios of our algorithms with the state-of-the-art algorithms.

To supplement the theoretical results, we conduct experiments across diverse application domains, such as influence maximization, vertex cover, and sensor placement. The empirical results demonstrate that our algorithms produce higher-quality solutions than the state-of-the-art evolutionary algorithms. The experiments also empirically demonstrate that, when allowed to run for a longer time, our algorithms perform better than the greedy-based algorithm with the same approximation ratio of $1/2$. As an implementation contribution, we show that the bloom filters [Bloom, 1970] can be used in evolutionary algorithms to avoid duplicate evaluations resulting in low memory and execution times.

1.2 Additional Related Works

The classical work of [Nemhauser *et al.*, 1978] presented a greedy algorithm that achieves a $(1 - 1/e)$ approximation ratio for the cardinality constraint problem and makes $\mathcal{O}(kn)$ calls to the monotone submodular function f. The works of [Qian *et al.*, 2015; Friedrich and Neumann, 2015] designed evolutionary algorithms for this problem using the Pareto Optimization framework while achieving the same approximation guarantee of $(1 - 1/e)$. The subsequent work of [Crawford, 2019] designed evolutionary algorithms that significantly reduced the runtime while achieving an approximation ratio of $(1 - 1/e - \epsilon)$. For the dynamic cost constraints, Pareto Optimization [Roostapour *et al.*, 2022] and its variant [Bian *et al.*, 2021] are proven to admit $1/2(1 - 1/e)$ approximation. The works in [Iyer and Bilmes, 2013; Padmanabhan *et al.*, 2023] considered the scenario where the function c is also submodular. A related problem to SMC is maximizing $f - c$, which has been studied in [Harshaw *et al.*, 2019; Jin *et al.*, 2021; Qian, 2021]. Evolutionary algorithms have been proposed for a few other variants of submodular optimization problems [Chen *et al.*, 2022; Do and Neumann, 2021].

Algorithm	Approximation Ratio
POMC ([Qian *et al.*, 2017])	$1/2(1 - 1/e)$
EAMC ([Bian *et al.*, 2020])	$1/2(1 - 1/e)$
EVO-SMC (this work)	$1/2$
ST-EVO-SMC (this work)	$1/2$

Table 1: Approximation ratios comparison with SOTAs for Problem 1.

2 Preliminaries

Given a ground set V of size n and a set function $f : 2^V \to \mathbb{R}$, f is monotone if for any subsets $S \subseteq T \subseteq V$, $f(T) \geq f(S)$. The *marginal gain* of adding an element $x \in V \setminus S$ into S is $f(\{x\} \mid S) \triangleq f(S \cup \{x\}) - f(S)$. We write $f(\{x\} \mid S) = f(x \mid S)$ for for brevity and assume that f is normalized, i.e., $f(\emptyset) = 0$. A function f is submodular if for any set $S \subseteq T \subseteq V$ and any element $x \in V \setminus T$, the marginal gain of the function value of adding x to S is at least the marginal gain in the function value of adding x to a larger set T. This property is referred to as *diminishing return*. Formally,

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T). \quad (1)$$

We consider a modular cost function $c : 2^V \to \mathbb{R}$ over a set $S \subseteq V$ where $c(S) = \sum_{s \in S} c(s)$. Note that a function is modular iff the equality of Eq. (1) holds. The *density* of an element e adding to set S is defined to be $\frac{f(e|S)}{c(e)}$. Define the maximal size of a feasible set as $K_\beta = \max\{|X| : X \subseteq V \wedge c(X) \leq \beta\}$.

Problem 1 (Submodular Maximization with Cost constraints (SMC))**.** *Given a monotone submodular function* $f : 2^V \to \mathbb{R}$ *and a modular cost function* $c : 2^V \to \mathbb{R}$*, a budget* β*, find a subset* $X \subset V$ *such that* $\arg\max_{X \subset V, c(X) \leq \beta} f(X)$.

The Chernoff bound is an exponentially decreasing upper bound on the tail of random variables. We present the form of the definition that will be used in this paper.

Lemma 1 (Chernoff Bound)**.** *Consider independent 0-1 variables* $Y_1, \cdots, Y_T$ *with the same expectations (means) and*

$Y = \sum_{i=1}^{T} Y_i$, if $\mathbb{E}[Y] = \mu$, for a real number $\eta \in (0,1)$, we have

$$\Pr[Y \leq (1-\eta)\mu] \leq e^{-\eta^2\mu/2}.$$

Oracles. In this paper, our algorithms are designed under the value oracle model, which is based on the assumption that there exists an oracle capable of returning the value $f(S)$ with a set S. The specific value of $f(S)$ can be the expected influence spread in influence diffusion [Kempe *et al.*, 2003], the covered vertices in directed vertex coverage, or entropy in sensor placement. Our computational complexity analyses and experiments are based on oracles.

3 An Evolutionary Algorithm EVO-SMC

In this section, we introduce an evolutionary algorithm EVO-SMC for Problem 1 along with its theoretical guarantee and analysis. We proceed with defining an auxiliary function g, referred to as *surrogate function* in [Bian *et al.*, 2020] that is useful in this algorithm.

Definition 1 (Surrogate function $g(X)$)**.** *For a set $X \subseteq V$,*

$$g(X) = \begin{cases} f(X)/c(X) & |X| \geq 1 \\ f(X) & |X| = 0. \end{cases}$$

The pseudocode of EVO-SMC is presented in Algorithm 1. The algorithm maintains three sets of candidate solutions F, G, G'. The i-th element of each F and G is of size i, and are denoted by F_i and G_i. G'_i is an augmentation of set G_i using a maximal marginal gain element (inspired by [Yaroslavtsev *et al.*, 2020]). The candidate solutions conform to the cost constraint for the problem. The algorithm takes an input T, the loop controlling parameter describing the number of iterations, and outputs the set for which the valuation of the objective function f is maximal.

At every iteration, the algorithm randomly selects a set S from the candidate solution sets F and G and "mutate" S (lines 3-4). The MUTATE procedure uniformly at random flips the membership of the elements in S with probability $1/n$. More specifically, for every element in V, if it appears in S, remove it from S with probability $1/n$, otherwise add it to S with probability $1/n$. If the mutated set S' satisfies the budget constraint, the algorithm compares S' with a set in the solution pools F and G with the *same* size as S'. The corresponding set(s) in the solution pools will be updated if S' is better w.r.t. g value or f value. The algorithm also considers an augmented solution G' with the largest marginal gain (lines 11-14), which will be necessary for the theoretical analysis.

Theorem 1. *Given a monotone submodular set function f, a modular cost function c, a cost constraint β, let $\mathrm{OPT}_\beta = \arg\max\{f(X) \mid X \subseteq V, c(X) \leq \beta\}$. If $K_\beta = \max\{|X| \mid X \subseteq V \wedge c(X) \leq \beta\}$, then after $T \geq \max\{4en^2K_\beta, 16en^2 \log n\}$ iterations, Algorithm 1 outputs X such that with probability $1 - \frac{1}{n}$,*

$$f(X) \geq 1/2 \cdot f(\mathrm{OPT}_\beta).$$

Algorithm 1: EVO-SMC

Input : $f : 2^V \to \mathbb{R}^+, c : 2^V \to \mathbb{R}^+$, total number of iterations $T \in \mathbb{Z}_{>0}$, cost constraint $\beta \in \mathbb{R}_{>0}$
Output: $\arg\max_{X \in \{F_0,\cdots,F_{n-1},G_0,\cdots,G_{n-1},G'_0,\cdots,G'_{n-1}\}} f(X)$

1 $F_j \leftarrow \emptyset, G_j \leftarrow \emptyset, G'_j \leftarrow \emptyset$ for all $j \in [0, n-1]$
2 **for** $t \leftarrow 1$ ***to*** T **do**
3 $\quad S \leftarrow$ Random($\{F_0, \cdots, F_{n-1}, G_0, \cdots, G_{n-1}\}$)
4 $\quad S' \leftarrow$ MUTATE (S)
5 $\quad i \leftarrow |S'|$
6 $\quad$ **if** $c(S') \leq \beta$ **then**
7 $\quad\quad$ **if** $f(F_i) < f(S')$ **then**
8 $\quad\quad\quad F_i \leftarrow S'$
9 $\quad\quad$ **end**
10 $\quad\quad$ **if** $g(G_i) < g(S')$ **then**
11 $\quad\quad\quad Q \leftarrow G_i \cup \{v\}$ s.t.
$$v = \arg\max_{e \in V \setminus G_i, c(e) \leq \beta - c(G_i)} f(e \mid G_i)$$
12 $\quad\quad\quad$ **if** $f(Q) > f(G'_i)$ **then**
13 $\quad\quad\quad\quad G'_i \leftarrow Q$
14 $\quad\quad\quad$ **end**
15 $\quad\quad\quad G_i \leftarrow S'$
16 $\quad\quad$ **end**
17 $\quad$ **end**
18 **end**

In order to prove the above theorem, we will introduce an auxiliary variable ω, which keeps track of a *good mutation*. We proceed with the definition of ω and followed by a necessary lemma for the proof of Theorem 1.

Definition 2 (Good mutation and ω)**.** *The valuation of ω at the end of iteration t is denoted by $\omega(t)$. We say that $\omega(0) = 0$, i.e., ω initially is 0. $\omega(t) = \omega(t-1) + 1$ is incremented iff the following two conditions are met during an iteration t*

1. *A specific set $S = G_{\omega(t-1)}$ is selected for mutation at line 3 of Algorithm 1.*
2. *There is exactly one element added to S during* MUTATE *procedure, nothing else, say $S' = S \cup \{a\}$, where $a = \arg\max_{e \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})} \frac{f(e|S)}{c(e)}$, where $o^* = \arg\max_{e \in \mathrm{OPT}_\beta} c(e)$.*

The above conditions collectively correspond to a good mutation. In other words, the valuation of ω is incremented at the end of iteration t iff a good mutation occurs.

At the end of iteration t, we define X_t to be the set where $X_t = G_{\omega(t)}$. Note that, if for iterations i and j such that $\omega(i) = \omega(j)$ and $i > j$, then $f(X_i) \geq f(X_j)$ and $g(X_i) \geq g(X_j)$, which are guaranteed by lines 7-14 of Algorithm 1.

Next, we present the lemma that forms the basis for the proof of the Theorem 1.

Lemma 2. *In Algorithm 1, at the beginning of an iteration t, if the set S is selected for mutation such that $c(S) \leq c(\mathrm{OPT}_\beta) - c(o^*)$ where $o^* = \arg\max_{e \in \mathrm{OPT}_\beta} c(e)$, and if the algorithm adds the element v to S where $v =$*

$\arg\max_{e \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})} \frac{f(e|S)}{c(e)}$, *then,*

$$f(v \mid S) \geq \frac{c(v)}{c(\mathrm{OPT}_\beta) - c(o^*)} \left(f(\mathrm{OPT}_\beta) - f(S \cup \{o^*\}) \right).$$

Proof. Since $v = \arg\max_{e \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})} \frac{f(e|S)}{c(e)}$, for every $u \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})$, we have

$$\frac{f(v \mid S)}{c(v)} \geq \frac{f(u \mid S)}{c(u)}. \tag{2}$$

Next consider the following inequalities.

$$\begin{aligned}
& f(\mathrm{OPT}_\beta) - f(S \cup \{o^*\}) \\
& \leq f(\mathrm{OPT}_\beta \cup S \cup \{o^*\}) - f(S \cup \{o^*\}) \\
& \leq \sum_{u \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})} f(u|S \cup \{o^*\}) \quad \textit{due to submodularity} \\
& \leq \sum_{u \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})} f(u|S) \qquad \textit{due to submodularity} \\
& \leq \sum_{u \in \mathrm{OPT}_\beta \setminus (S \cup \{o^*\})} \frac{f(v|S)}{c(v)} \cdot c(u) \qquad \textit{from Eq. (2)} \\
& \leq \frac{f(v|S)}{c(v)} \left(c(\mathrm{OPT}_\beta) - c(o^*) \right).
\end{aligned}$$

Rearranging the terms concludes the proof of Lemma 2. □

To finish the approximation guarantee proof of Theorem 1, we next prove the following lemma.

Lemma 3. *For any iteration $t \in [1, T]$, let X_t be the set with size $\omega(t)$ at the end of iteration t, then we have:*

$$f(X_t) \geq \frac{c(X_t)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta).$$

Proof. We prove this lemma by regular induction.

Base Case: For the first iteration $t = 1$ of Algorithm 1, we have two cases B-1 and B-2.

Case B-1: ω is not incremented, then we have $\omega(1) = \omega(0) = 0$. By the definition of X_t, $|X_0| = |X_1| = \omega(1) = 0$ and $X_1 = \emptyset$. So X_1 holds for Lemma 3 by the following inequality.

$$f(X_1) = 0 \geq \frac{c(X_1)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) = 0.$$

Case B-2: ω is incremented, then we have $\omega(1) = \omega(0) + 1 = 1$. Since this is the first iteration, $X_0 = \emptyset$, $|X_1| = 1$. Assume that the element in condition 2 of Definition 2 is v. Then, $X_1 = \{v\}$, and by applying Lemma 2, we have

$$\begin{aligned}
f(X_1) &= f(X_1) - f(X_0) = f(\{v\}) - f(\emptyset) \\
&\geq \frac{c(v)}{c(\mathrm{OPT}_\beta) - c(o^*)} (f(\mathrm{OPT}_\beta) - f(\{o^*\})).
\end{aligned}$$

If $f(\{o^*\}) \geq \frac{1}{2} f(\mathrm{OPT}_\beta)$, then as per the lines 11-14 of Algorithm 1, it is guaranteed to admit a $1/2$ approximation. Therefore, we focus on the case where $f(\{o^*\}) < \frac{1}{2} f(\mathrm{OPT}_\beta)$. Using this relation, we have

$$\begin{aligned}
f(X_1) &\geq \frac{c(v)}{(c(\mathrm{OPT}_\beta) - c(o^*))} \times \frac{f(\mathrm{OPT}_\beta)}{2} \\
&= \frac{c(X_1)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) \quad \textit{due to } X_1 = \{v\}.
\end{aligned}$$

Thus, Lemma 3 holds for the base case ($t = 1$). Next, we prove that for iterations $t \in (1, T]$, Lemma 3 still holds.

Induction Steps: For iterations $t > 1$, the induction hypothesis (I.H.) is: at the end of iteration $t - 1$, we have

$$f(X_{t-1}) \geq \frac{c(X_{t-1})}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta). \tag{3}$$

For every iteration $t > 1$, we analyze X_t. We also have two cases I-1 and I-2 based on whether ω is incremented.

Case I-1: If ω is not incremented, then we have $\omega(t) = \omega(t-1)$, and $|X_t| = |X_{t-1}| = \omega(t-1)$. Furthermore, the Algorithm 1 (lines 7-14) ensures that $g(X_t) \geq g(X_{t-1})$. Therefore, by Definition 1, we can derive

$$g(X_t) \geq g(X_{t-1}) \Rightarrow \frac{f(X_t)}{c(X_t)} \geq \frac{f(X_{t-1})}{c(X_{t-1})}. \tag{4}$$

Rearranging the above inequality, we have

$$\begin{aligned}
& f(X_t) \geq \frac{c(X_t)}{c(X_{t-1})} f(X_{t-1}) \qquad \textit{due to Eq. (4)} \\
& \geq \frac{c(X_t)}{c(X_{t-1})} \cdot \frac{c(X_{t-1})}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) \;\; \textit{due to I.H.} \\
& = \frac{c(X_t)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta).
\end{aligned}$$

This concludes the proof of Lemma 3 for Case I-1.

Case I-2: If ω is incremented, then we have $\omega(t) = \omega(t-1) + 1$. In this case, X_{t-1} is selected for mutation and the *good* mutation results in $|X_t| = |X_{t-1} \cup \{v\}|$ where $v = \arg\max_{u \in \mathrm{OPT}_\beta \setminus (X_{t-1} \cup \{o^*\})} \{ \frac{f(u|X_{t-1})}{c(u)} \}$.

• *Case I-2.1:* $c(X_{t-1}) \geq c(\mathrm{OPT}_\beta) - c(o^*)$

We argue that, in this case, the algorithm already has admitted a $1/2$-approximation at the end of iteration $t - 1$.

$$\begin{aligned}
f(F_{\omega(t)}) &\geq f(X_{t-1} \cup \{v\}) \geq f(X_{t-1}) \\
&\geq \frac{c(X_{t-1})}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) \qquad \textit{due to I.H.} \\
&\geq \frac{1}{2} f(\mathrm{OPT}_\beta).
\end{aligned}$$

• *Case I-2.2:* $c(X_{t-1}) < c(\mathrm{OPT}_\beta) - c(o^*)$.

We proceed with the following claim which can be proved by considering the cases of $o^* \in X_{t-1}$ and $o^* \notin X_{t-1}$.

Claim 1. *WLOG, for any iteration $t < T$, when $c(X_{t-1}) < c(\mathrm{OPT}_\beta) - c(o^*)$ we claim that*

$$f(X_{t-1} \cup \{o^*\}) < 1/2 f(\mathrm{OPT}_\beta). \tag{5}$$

Otherwise, at the end of iteration $t - 1$, the algorithm has already admitted a $1/2$-approximation.

Going back to the proof for Case I-2 of the Lemma 3, as per its condition: $c(X_{t-1}) < c(\mathrm{OPT}_\beta) - c(o^*)$. By rearranging Lemma 2 with $S = X_{t-1}$, we have:

$$
\begin{aligned}
f(X_{t-1} \cup \{v\}) &\geq \frac{c(v)}{c(\mathrm{OPT}_\beta) - c(o^*)} f(\mathrm{OPT}_\beta) + f(X_{t-1}) \\
&\quad - \frac{c(v)}{c(\mathrm{OPT}_\beta) - c(o^*)} f(X_{t-1} \cup \{o^*\}) \\
&\geq \frac{c(v)}{c(\mathrm{OPT}_\beta) - c(o^*)} f(\mathrm{OPT}_\beta) + f(X_{t-1}) \\
&\quad - \frac{c(v)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) \quad \textit{due to Claim 1} \\
&\geq \frac{c(v)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) \\
&\quad + \frac{c(X_{t-1})}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta) \quad \textit{due to I.H.} \\
&= \frac{c(X_{t-1} \cup \{v\})}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta).
\end{aligned}
$$

Now observe that for a specific cardinality of the parameter, the function $g(\cdot)$ is non-decreasing. Therefore,

$$
g(X_t) \geq g(X_{t-1} \cup \{v\}) \Rightarrow \frac{f(X_t)}{c(X_t)} \geq \frac{f(X_{t-1} \cup \{v\})}{c(X_{t-1} \cup \{v\})}.
$$

Thus, using the above inequalities, we have:

$$
f(X_t) \geq \frac{c(X_t) \cdot f(X_{t-1} \cup \{v\})}{c(X_{t-1} \cup \{v\})} \geq \frac{c(X_t) \cdot f(\mathrm{OPT}_\beta)}{2(c(\mathrm{OPT}_\beta) - c(o^*))}.
$$

This concludes the proof for Lemma 3. □

Justification for 1/2 - approximation. If at the end of some iteration t', $c(X_{t'}) \geq c(\mathrm{OPT}_\beta) - c(o^*)$, then by Lemma 3, EVO-SMC admits 1/2 - approximation.

If there does not exist such t', we consider the last iteration t where ω is incremented. Similar to the proof of Case I-2 in Lemma 3, we analyze $c(X_{t-1} \cup \{v\})$. Note that, ω is incremented for a good mutation where $v = \arg\max_{e \in \mathrm{OPT}_\beta \setminus (X_{t-1} \cup \{o^*\})} \{\frac{f(e|X_{t-1})}{c(e)}\}$. In that case, we can claim $c(X_{t-1} \cup \{v\}) \leq \beta$, which follows from

$$
\begin{aligned}
&c(X_{t-1}) < c(\mathrm{OPT}_\beta) - c(o^*) \leq c(\mathrm{OPT}_\beta) - c(v) \leq \beta - c(v) \\
&\quad \Rightarrow c(X_{t-1} \cup \{v\}) \leq \beta.
\end{aligned}
$$

(Recall that o^* is the element in OPT_β with the maximal cost.)

After such a good mutation, if $c(X_{t-1} \cup \{v\}) \geq c(\mathrm{OPT}_\beta) - c(o^*)$, then the algorithm will output the optimal solution. The reason is that $\omega(t)$ can be up to n, i.e., the potential solution set size $|X_t| = |X_{t-1} \cup \{v\}|$ can be as large as n. Otherwise, the algorithm can keep running until $c(X_{t-1} \cup \{v\}) \geq c(\mathrm{OPT}_\beta) - c(o^*)$.

Bound on the Number of Iterations. We analyze the running time of Algorithm 1 when it achieves a 1/2 approximation. The following Observation 1 bounds the probability of a good mutation and Lemma 4 concludes the analysis.

Observation 1. *ω is incremented with a probability at least $\frac{1}{2en^2}$ at the end of every iteration.*

Lemma 4. *In Algorithm 1, define variable Y_i for iteration $i \in [1, T]$ where $Y_i = 1$ if $\omega(i) = \omega(i-1) + 1$ and $Y_i = 0$ otherwise. If $T \geq \max\{4en^2 K_\beta, 16en^2 \log n\}$, then*

$$
\Pr\left[\sum_{i=1}^{T} Y_i < K_\beta\right] \leq \frac{1}{n}.
$$

Proof. Observation 1 lower bounds the probability of $Y_i = 1$ for an iteration i. Thus, $\mu = \mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=1}^{T} Y_i\right] = T\rho \geq T \cdot \frac{1}{2en^2}$. We can apply Chernoff bound as presented in Lemma 1 to claim the following inequalities.

$$
\Pr\left[\sum_{i=1}^{T} Y_i < K_\beta\right] \leq \Pr\left[\sum_{i=1}^{T} Y_i < T\rho/2\right] \leq e^{-T\rho/8} \leq \frac{1}{n}.
$$

The first inequality holds if we take $T \geq 4en^2 K_\beta$. The second inequality follows from the Chernoff bound with $\eta = 1/2$. The last inequality holds because $T \geq 16en^2 \log n$. □

Lemmas 3 and 4 conclude the proof of Theorem 1.

4 Stochastic Evolutionary Algorithm ST-EVO-SMC

In this section, we correlate the idea of biased technique from [Crawford, 2019] and design a faster algorithm ST-EVO-SMC for Problem 1.

The algorithm (Algorithm 2) has input parameters $\epsilon \in (0, 1]$ and $p \in (0, 1]$ to allow users to balance the approximation guarantee and running time. The larger ϵ is, the lower the approximation the algorithm admits, and fewer iterations are required. One can notice that ST-EVO-SMC is deducted to Algorithm 1 if $p = 0$. On the other hand, if p is very large, the algorithm becomes a sampling-based stochastic algorithm similar to [Mirzasoleiman *et al.*, 2015]. The given parameter p controls the stochasticity probability of finding a "good" candidate set in the first condition of Definition 2. In Algorithm 1, the probability of randomly selecting a set for mutation is $\frac{1}{2n}$. However, in ST-EVO-SMC, the probability is at least p. Therefore, Algorithm 2 can grow ω rapidly and results in smaller amount of iterations.

Theorem 2. *With a monotone submodular function f, modular cost function c, cost constraint β, the maximal seed set size K_β, error threshold $\epsilon \in (0, 1]$ and stochasticity probability $p \in (0, 1]$, let $\mathrm{OPT}_\beta = \arg\max_{X \subseteq V, c(X) \leq \beta}\{f(X)\}$. After $T \geq 2enK_\beta \ln(1/\epsilon)/p$ iterations, Algorithm 2 outputs X and admits $\frac{1}{2}$- approximation with probability $1 - \epsilon$.*

Proof Sketch: The idea is to bond the increment of ω (line 9) with a good mutation. If ω is incremented, then G_ω at line 6 has been selected for mutation for at least H time, and it is very likely that there ever exists at least one good mutation during those mutations. If we define E_t: ω was incremented at the end of iteration t; F_t: Given E_t, there exists at least one good mutation during some iteration(s) since the last iteration ω was incremented. We have Claim 2 showing that F_t happens with probability $1 - \epsilon$ given that E_t happens.

Algorithm 2: ST-EVO-SMC

Input : $f : 2^V \to \mathbb{R}^+, c : 2^V \to \mathbb{R}^+$
$T \in \mathbb{Z}_{>0}, \beta \in \mathbb{R}_{>0}, \epsilon \in (0, 1], p \in [0, 1]$
Output: $\arg\max_{X \in \{F_0,\cdots,F_{n-1},G_0,\cdots,G_{n-1},G'_0,\cdots,G'_{n-1}\}} f(X)$

1 $F_j \leftarrow \emptyset, G_j \leftarrow \emptyset, G'_j \leftarrow \emptyset$, for all $j \in [0, n-1]$
2 $\omega \leftarrow 0, \ell \leftarrow 1, H \leftarrow \lceil en \log(1/\epsilon) \rceil$
3 **for** $t \leftarrow 1$ ***to*** T **do**
4     $S \leftarrow$ Random($\{F_0, \cdots, F_{n-1}, G_0, \cdots, G_{n-1}\}$)
5   **if** FLIP-COIN(p) $=$ *heads* **then**
6         $S \leftarrow G_\omega$
7         $\ell \leftarrow \ell + 1$
8   **if** $\ell \,\%\, H = 0$ **then**
9             $\omega \leftarrow \omega + 1$
10   **end**
11   **end**
12     $S' \leftarrow$ MUTATE (S)
13     $i \leftarrow |S'|$
14   **if** $c(S') \leq \beta$ **then**
15   **if** $f(F_i) < f(S')$ **then**
16             $F_i \leftarrow S'$
17   **end**
18   **if** $g(G_i) < g(S')$ **then**
19             $Q \leftarrow G_i \cup \{v\}$ s.t.

$$v = \arg\max_{e \in V \setminus G_i, c(e) \leq \beta - c(G_i)} f(e \mid G_i)$$

20   **if** $f(Q) > f(G'_i)$ **then**
21                 $G'_i \leftarrow Q$
22   **end**
23             $G_i \leftarrow S'$
24   **end**
25   **end**
26 **end**

Claim 2. $\Pr[F_t \mid E_t] \geq 1 - \epsilon$.

Analogous to the proof of Theorem 1, we need the following lemma to guarantee a $1/2$-approximation ratio.

Lemma 5. *For every iteration t such that $1 \leq t \leq T$, let X_t be the set with size $\omega(t)$ at the end of iteration t, then with probability $1 - \epsilon$, we have:*

$$f(X_t) \geq \frac{c(X_t)}{2(c(\mathrm{OPT}_\beta) - c(o^*))} f(\mathrm{OPT}_\beta). \tag{6}$$

Proof Sketch: The proof uses strong induction proof methods. The base cases include iterations with $\omega = 0$ at time and the first iteration with $\omega = 1$. Conditioned on $\neg E_t$, Lemma 3 can be verified; when F_t occurs, applying Lemma 2 will finish the proof. For inductive steps, the inductive hypothesis is: Eq. (6) is true for all iterations i such that $1 \leq i < t$. The proof follows similar to Case I-2 proof in Section 3.

The following lemma ensures that ω has been incremented for K_β times after T iterations with probability $1 - \epsilon$.

Lemma 6. *In Algorithm 2, define $0-1$ variables Y_i for iteration $i \in [1, T]$ where $T \geq 2enK_\beta \ln(1/\epsilon)/p$. $Y_i = 1$ if ℓ is incremented at line 7 of Algorithm 2, 0 otherwise. Then,*

$$\Pr\left[\sum_{i=1}^{T} Y_i < HK_\beta\right] \leq \epsilon.$$

5 Experiments

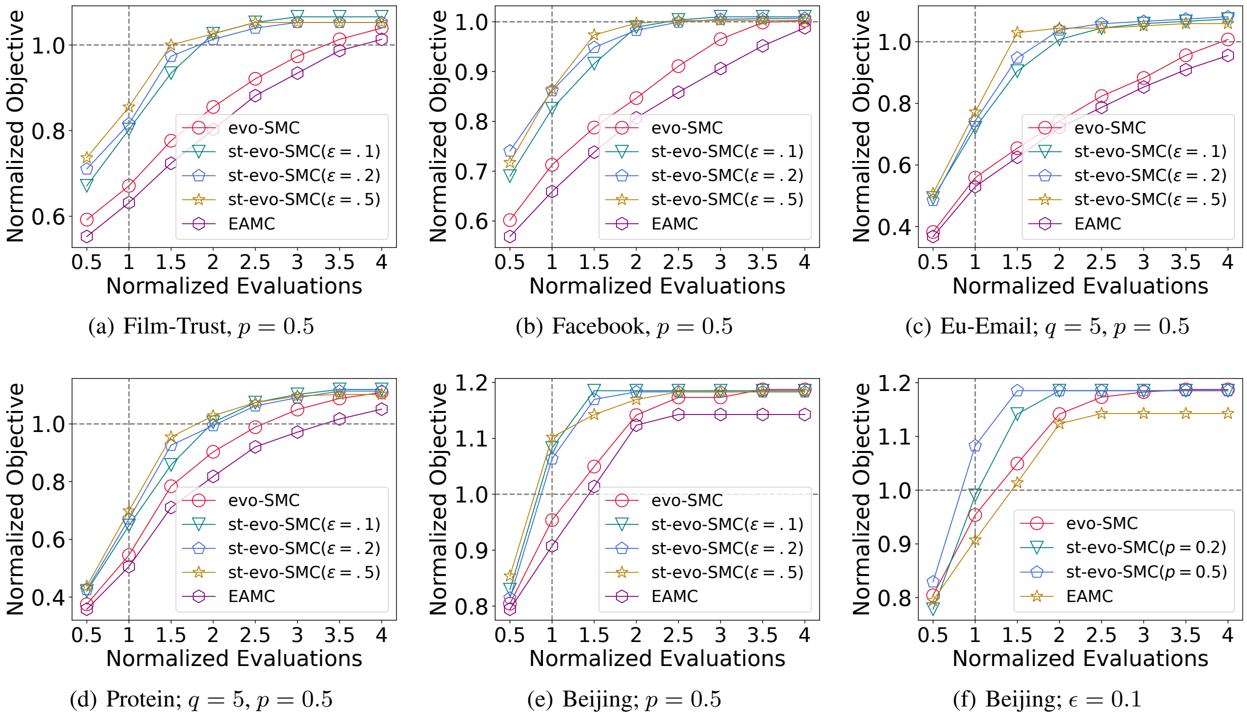
EVO-SMC and ST-EVO-SMC are evaluated on influence maximization, directed vertex cover, and sensor placement with costs. We compare our algorithms with the evolutionary EAMC [Bian *et al.*, 2020] and the deterministic algorithm Greedy+Max [Yaroslavtsev *et al.*, 2020] (1/2-approximation and runs in $\mathcal{O}(nK_\beta)$). We implement our algorithms and baselines in C++ (`https://github.com/yz24/evo-SMC`). We run our algorithms and EAMC 20 times and report the medians.

Implementation Accelerations. The mutation procedures in algorithms EVO-SMC, ST-EVO-SMC and EAMC are performed independently at every iteration. Therefore, the set after mutation S' can stay the same with non-trivial probability or be equal to some set that has already been considered. Thus, to avoid repeated evaluations of the objective f and g over the same set, we maintain a bloom filter [Bloom, 1970] and pre-check the incoming set S'. In our implementations and results, the number of evaluations of f is incremented only if an evaluation of $f(S')$ is actually performed.

5.1 Applications and Experimental Settings

Influence Maximization with Costs. Influence maximization in a social network $G = (V, E)$ seeks to maximize $\mathbb{E}[\mathbf{IC}(X)]$, the expected number of influenced users influenced by the propagation of information from a subset of users (seed set) $X \subseteq V$. The Independent Cascade (IC) model estimates $\mathbb{E}[\mathbf{IC}(X)]$ with propagation probabilities $p(u, v)$ [Kempe *et al.*, 2003]. In real-life scenarios, there is a total budget β, and each node $v \in V$ has a cost $c(v)$ (can be viewed as an incentive) in the propagation process. Hence, the cost-constrained influence maximization problem can be formulated as: finding X such that $\arg\max_{X \subseteq V, c(X) \leq \beta} \mathbb{E}[\mathbf{IC}(X)]$. We define a linear cost function $c : V \to \mathbb{R}^+$ proportional to the out-degree $out(v)$ [Jin *et al.*, 2021], where $c(v) = \lambda \cdot out(v)^\gamma$ with free parameters γ and λ. If $out(v) = 0$, $c(v)$ is set to 1. In our experiments, we use Facebook [Leskovec and Mcauley, 2012] and Film-Trust networks [Kunegis, 2013]. We run the algorithms with fixed budget $\beta = 20$ and generate node costs with $\lambda = 1.2, \gamma = 1.5$.

Directed Vertex Cover with Costs. Let $G = (V, E)$ be a directed graph and $w : 2^V \to \mathbb{R}^{\geq 0}$ be a modular weight function on a subset of vertices. For a vertex set $S \subseteq V$, let $N(S)$ denote the set of vertices which are pointed to by S, formally, $N(S) = \{v \in V \mid (u, v) \in E \wedge u \in S\}$. The weighted directed vertex cover function is $f(S) = \sum_{u \in N(S) \cup S} w(u)$, which is monotone submodular. We also assume that each vertex $v \in V$ has an associated non-negative modular cost function $c(v)$ [Harshaw *et al.*, 2019] defined by $c(v) = 1 + \max\{d(v) - q, 0\}$, where $d(v)$ is the out-degree of vertex v and the non-negative integer q is the cost penalty. The objective of this task is to find a subset S such that



(a) Film-Trust, $p = 0.5$ (b) Facebook, $p = 0.5$ (c) Eu-Email; $q = 5, p = 0.5$

(d) Protein; $q = 5, p = 0.5$ (e) Beijing; $p = 0.5$ (f) Beijing; $\epsilon = 0.1$

Figure 1: Experimental results with various applications.

$\arg\max_{S \subseteq V, c(S) \le \beta} \sum_{u \in N(S) \cup S} w(u)$. We use Protein network [Stelzl *et al.*, 2005] and Eu-Email network [Leskovec *et al.*, 2007] in this application. We assign each node a weight of 1 and generate costs as mentioned above. We report results of $\beta = 30$ and cost penalty $q = 5$ for both networks.

Sensor Placement with Costs. We use a real-world air quality data (light and temperature measures) [Zheng *et al.*, 2013] [Bian *et al.*, 2020] collected from 36 monitoring stations in Beijing. We calculate the entropy of a sensor placement using the observed frequency. Each sensor is assigned a positive cost from the normal distribution $\mathcal{N}(0, 1)$. We maximize the submodular Entropy function with a budget $\beta = 10$.

5.2 Results

The comparison results on influence maximization (Film-Trust and Facebook), vertex cover (Protein and Eu-Email) and sensor placement (Beijing) are illustrated in Figures 1(a), 1(b), Figures 1(c), 1(d) and Figures 1(e), 1(f), respectively. Note that the objective values and runtime are normalized by the objective value and number of oracle calls made ($\mathcal{O}(nK_\beta)$) by the Greedy+Max algorithm. The grey lines in the figures are 1's, corresponding to the objective value and oracle evaluations of Greedy+Max.

Due to the accelerations with bloom filters and pre-check step, our algorithm EVO-SMC is as efficient as EAMC. Moreover, the stochastic version algorithm ST-EVO-SMC consistently performs better than EAMC and outperforms Greedy+Max after making approximately two times oracle evaluations of Greedy+Max (exceeding the grey horizontal lines). As expected, the objective values of EVO-SMC grow slowly compared to ST-EVO-SMC. The reason is that EVO-SMC randomly selects a set to mutate while ST-EVO-SMC performs the selection process with a purpose (with a stochasticity probability p). We can also observe that when $\epsilon = 0.1$, ST-EVO-SMC produces higher-quality results than with larger error thresholds, which is consistent with the theoretical guarantees. In Figure 1(e), we plot the results with various p choices. With a larger $p = 0.5$, ST-EVO-SMC has a faster initial increase compared with ST-EVO-SMC($p = 0.2$).

During the experiments, we observe that the valuation of cost of partial solutions accumulates to β quickly and then stay close to β. This means there were a very small portion or no elements to evaluate to find the augmented element (line 19 of Algorithm 2), which is a reason why EVO-SMC is as efficient as EAMC.

6 Conclusions

In this paper, we proposed novel evolutionary frameworks for submodular maximization with cost constraints. Our algorithms achieve competitive approximation guarantees compared to the state-of-the-art evolutionary methods. Empirical studies demonstrate that our algorithms are also efficient. Future work would be designing faster algorithms that can efficiently adapt to the dynamic cost constraint settings. Unlike EAMC [Bian *et al.*, 2020], the surrogate function defined in our paper is independent of the budget β. Therefore, it is promising to investigate how our framework can deal with budget changes in dynamic cost settings.

Acknowledgements

Part of Pavan's work is supported by NSF grant 2130536.

References

[Badanidiyuru and Vondrák, 2014] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1497–1514. SIAM, 2014.

[Bateni *et al.*, 2019] MohammadHossein Bateni, Lin Chen, Hossein Esfandiari, Thomas Fu, Vahab Mirrokni, and Afshin Rostamizadeh. Categorical feature compression via submodular optimization. In *International Conference on Machine Learning*, pages 515–523. PMLR, 2019.

[Bian *et al.*, 2020] Chao Bian, Chao Feng, Chao Qian, and Yang Yu. An efficient evolutionary algorithm for subset selection with general cost constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3267–3274, 2020.

[Bian *et al.*, 2021] Chao Bian, Chao Qian, Frank Neumann, and Yang Yu. Fast pareto optimization for subset selection with dynamic cost constraints. In *IJCAI*, pages 2191–2197, 2021.

[Bloom, 1970] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[Chen *et al.*, 2022] Xuefeng Chen, Liang Feng, Xin Cao, Yifeng Zeng, and Yaqing Hou. Budgeted sequence submodular maximization. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4733–4739. ijcai.org, 2022.

[Crawford, 2019] Victoria G. Crawford. An efficient evolutionary algorithm for minimum cost submodular cover. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1227–1233. ijcai.org, 2019.

[Do and Neumann, 2021] Anh Viet Do and Frank Neumann. Pareto optimization for subset selection with dynamic partition matroid constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12284–12292, 2021.

[El Halabi *et al.*, 2022] Marwa El Halabi, Suraj Srinivas, and Simon Lacoste-Julien. Data-efficient structured pruning via submodular optimization. *Advances in Neural Information Processing Systems*, 35:36613–36626, 2022.

[Ene and Nguyen, 2019] Alina Ene and Huy L. Nguyen. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 53:1–53:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Feldman *et al.*, 2022] Moran Feldman, Zeev Nutov, and Elad Shoham. Practical budgeted submodular maximization. *Algorithmica*, pages 1–40, 2022.

[Friedrich and Neumann, 2015] Tobias Friedrich and Frank Neumann. Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evolutionary computation*, 23(4):543–558, 2015.

[Harshaw *et al.*, 2019] Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *International Conference on Machine Learning*, pages 2634–2643. PMLR, 2019.

[Iyer and Bilmes, 2013] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. *Advances in neural information processing systems*, 26, 2013.

[Jin *et al.*, 2021] 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. *Proceedings of the VLDB Endowment*, 14(10):1756–1768, 2021.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

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

[Krause and Guestrin, 2005] Andreas Krause and Carlos Guestrin. *A note on the budgeted maximization of submodular functions*. Citeseer, 2005.

[Kunegis, 2013] Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*, pages 1343–1350, 2013.

[Leskovec and Mcauley, 2012] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[Leskovec *et al.*, 2007] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[Li *et al.*, 2022] Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. In *Advances in Neural Information Processing Systems*, 2022.

[Li *et al.*, 2023] Ruolin Li, Negar Mehr, and Roberto Horowitz. Submodularity of optimal sensor placement for traffic networks. *Transportation Research Part B: Methodological*, 171:29–43, 2023.

[Mirzasoleiman *et al.*, 2015] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and

Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

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

[Padmanabhan *et al.*, 2023] Madhavan R Padmanabhan, Yanhui Zhu, Samik Basu, and A Pavan. Maximizing submodular functions under submodular constraints. In *Uncertainty in Artificial Intelligence*, pages 1618–1627. PMLR, 2023.

[Qian *et al.*, 2015] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. *Advances in neural information processing systems*, 28, 2015.

[Qian *et al.*, 2017] Chao Qian, Jing-Cheng Shi, Yang Yu, and Ke Tang. On subset selection with general cost constraints. In *IJCAI*, volume 17, pages 2613–2619, 2017.

[Qian, 2021] Chao Qian. Multiobjective evolutionary algorithms are still good: Maximizing monotone approximately submodular minus modular functions. *Evolutionary Computation*, 29(4):463–490, 2021.

[Roostapour *et al.*, 2022] Vahid Roostapour, Aneta Neumann, Frank Neumann, and Tobias Friedrich. Pareto optimization for subset selection with dynamic cost constraints. *Artificial Intelligence*, 302:103597, 2022.

[Stelzl *et al.*, 2005] Ulrich Stelzl, Uwe Worm, Maciej Lalowski, Christian Haenig, Felix H Brembeck, Heike Goehler, Martin Stroedicke, Martina Zenkner, Anke Schoenherr, Susanne Koeppen, et al. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968, 2005.

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

[Tang *et al.*, 2020] Jing Tang, Xueyan Tang, Andrew Lim, Kai Han, Chongshou Li, and Junsong Yuan. Revisiting modified greedy algorithm for monotone submodular maximization with a knapsack constraint. *CoRR*, abs/2008.05391, 2020.

[Yaroslavtsev *et al.*, 2020] Grigory Yaroslavtsev, Samson Zhou, and Dmitrii Avdiukhin. “bring your own greedy”+ max: near-optimal 1/2-approximations for submodular knapsack. In *International Conference on Artificial Intelligence and Statistics*, pages 3263–3274. PMLR, 2020.

[Zheng *et al.*, 2013] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444, 2013.

[Zhu *et al.*, 2024] Yanhui Zhu, Fang Hu, Lei Hsin Kuo, and Jia Liu. Scoreh+: A high-order node proximity spectral clustering on ratios-of-eigenvectors algorithm for community detection. *IEEE Transactions on Big Data*, 10(3):301–312, 2024.