Fast Pareto Optimization for Subset Selection with Dynamic Cost Constraints

Chao Bian[1], **Chao Qian**[1*], **Frank Neumann**[2] and **Yang Yu**[1]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]Optimisation and Logistics, The University of Adelaide, Adelaide SA 5005, Australia
{bianc,qianc,yuy}@lamda.nju.edu.cn, frank.neumann@adelaide.edu.au

Abstract

Subset selection with cost constraints is a fundamental problem with various applications such as influence maximization and sensor placement. The goal is to select a subset from a ground set to maximize a monotone objective function such that a monotone cost function is upper bounded by a budget. Previous algorithms with bounded approximation guarantees include the generalized greedy algorithm, POMC and EAMC, all of which can achieve the best known approximation guarantee. In real-world scenarios, the resources often vary, i.e., the budget often changes over time, requiring the algorithms to adapt the solutions quickly. However, when the budget changes dynamically, all these three algorithms either achieve arbitrarily bad approximation guarantees, or require a long running time. In this paper, we propose a new algorithm FPOMC by combining the merits of the generalized greedy algorithm and POMC. That is, FPOMC introduces a greedy selection strategy into POMC. We prove that FPOMC can maintain the best known approximation guarantee efficiently.

1 Introduction

In this paper, we consider the subset selection problem with general cost constraints, i.e.,

$$\arg\max_{X\subseteq V} f(X) \quad s.t. \quad c(X) \le B, \tag{1}$$

where both the objective function $f : 2^V \to \mathbb{R}$ and the cost function $c : 2^V \to \mathbb{R}$ are monotone, but not necessarily submodular. This problem is NP-hard in general, and has various applications, such as influence maximization [Kempe *et al.*, 2003], sensor placement [Krause *et al.*, 2008], document summarization [Lin and Bilmes, 2011] and unsupervised feature selection [Feng *et al.*, 2019], just to name a few.

A well-known special case of this problem is subset selection with cardinality constraints, that is, $c(X) = |X|$. Das and Kempe [2018] proved that the simple greedy algorithm, which iteratively selects one item from V with the largest marginal gain on f, can achieve the optimal polynomial-time approximation guarantee of $(1 - e^{-\gamma_f})$ [Harshaw *et al.*, 2019], where γ_f measures how close f is to submodularity.

For the general problem Eq. (1), the simple greedy algorithm fails to obtain a bounded approximation ratio [Khuller *et al.*, 1999], while the generalized greedy algorithm can achieve the best known approximation ratio of $(\alpha_f/2)(1 - e^{-\alpha_f})$ [Zhang and Vorobeychik, 2016; Qian *et al.*, 2017], where α_f, different from γ_f, is another notion for measuring the closeness of f to submodularity. Unlike the simple greedy algorithm, the generalized greedy algorithm iteratively selects one item with the largest ratio of the marginal gain on f and c.

As the greedy behavior may limit the performance, Qian *et al.* [2017] proposed an anytime algorithm POMC by employing a multi-objective evolutionary algorithm (EA) to maximize the objective f and minimize the cost c simultaneously. POMC can find better solutions using more running time in practice. Though it can also achieve the $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio theoretically, the running time is not polynomially bounded. Thus, Bian *et al.* [2020] proposed another anytime algorithm EAMC by employing a single-objective EA to maximize the surrogate objective $f(X)/(1 - e^{-\alpha_f c(X)/B})$. EAMC achieves empirical performance competitive with POMC, but can guarantee the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio in polynomial running time.

In real-world applications of subset selection, the budget B on the cost constraint is, however, not fixed, but may change over time, reflecting the change of resources. For examples, in influence maximization, more market investment will make B increase; in sensor placement, some installed sensors may not work due to aging and thus B decreases. A natural question is then whether the generalized greedy algorithm, POMC and EAMC can adapt their solutions quickly when the budget B changes dynamically.

The answer is unfortunately negative. For the generalized greedy algorithm, Roostapour *et al.* [2019] have constructed instances where the approximation ratios are arbitrarily bad. Though POMC can maintain the $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio, the required running time depends on the population size, which is unbounded and may be even exponential [Roostapour *et al.*, 2019]. EAMC uses a surrogate

[*]This work was supported by the National Key Research and Development Program of China (2020AAA0107200), the Fundamental Research Funds for the Central Universities (0221-14380009), the NSFC (62022039, 61876077) and the Australian Research Council (DP160102401). Chao Qian is the corresponding author.

objective depending on B, and thus, the change of B leads to the change on the objective, making the found solutions probably unhelpful for solving the problem with a new B.

In this paper, we propose a new algorithm FPOMC for subset selection with dynamic cost constraints, i.e., dynamically changing B. As the generalized greedy algorithm runs efficiently but cannot guarantee the approximation ratio, while POMC maintains the approximation ratio but cannot guarantee the polynomial running time, the idea of FPOMC is naturally to combine their merits. In particular, FPOMC is modified from POMC by introducing a greedy selection strategy. POMC uses the uniform selection strategy, i.e., selects a solution from the population uniformly at random, for mutation, while FPOMC first selects a subset size uniformly at random, and then selects a specific solution with this size greedily from the population. We prove that when the budget B decreases, FPOMC has already achieved the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio; when B increases to B', FPOMC can regain the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio using at most $O(nK_{B'}(K_{B'}-K_B))$ expected running time, where K_B and $K_{B'}$ denote the largest size of a subset satisfying the two constraints $c(X) \leq B$ and $c(X) \leq B'$, respectively.

2 Subset Selection with Cost Constraints

Let $\mathbb{R}$ and $\mathbb{R}^+$ denote the set of reals and non-negative reals, respectively. Let $V = \{v_1, v_2, \dots, v_n\}$ denote a ground set. A set function $f : 2^V \to \mathbb{R}$ is monotone if

$$\forall X \subseteq Y : f(X) \leq f(Y).$$

A set function f is submodular if

$$\forall X \subseteq Y, v \notin Y : f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y),$$

implying the diminishing returns property [Nemhauser *et al.*, 1978]. The submodularity ratio

$$\alpha_f = \min_{X \subseteq Y, v \notin Y} \frac{f(X \cup \{v\}) - f(X)}{f(Y \cup \{v\}) - f(Y)} \qquad (2)$$

is used to characterize the closeness of a general set function f to submodularity [Zhang and Vorobeychik, 2016; Qian *et al.*, 2018]. When f satisfies the monotone property, we have $0 \leq \alpha_f \leq 1$, and f is submodular iff $\alpha_f = 1$.

As presented in Definition 1, the subset selection problem with static cost constraints is to maximize a monotone objective function f such that a monotone cost function c is no larger than a budget B. We assume w.l.o.g. that monotone functions are normalized, i.e., $f(\emptyset) = 0$ and $c(\emptyset) = 0$.

Definition 1 (Subset Selection with Static Cost Constraints)**.** *Given a monotone objective function $f : 2^V \to \mathbb{R}^+$, a monotone cost function $c : 2^V \to \mathbb{R}^+$ and a budget B, to find*

$$\arg\max\nolimits_{X \subseteq V} f(X) \quad \textit{s.t.} \quad c(X) \leq B. \qquad (3)$$

Because the cost function c is hard to be computed exactly in some real-world applications [Zhang and Vorobeychik, 2016; Qian *et al.*, 2017], we assume that only an $\psi(n)$-approximation $\hat{c}$ can be obtained, where

$$\forall X \subseteq V : c(X) \leq \hat{c}(X) \leq \psi(n) \cdot c(X). \qquad (4)$$

This general problem has many applications such as influence maximization [Kempe *et al.*, 2003] and sensor placement [Krause *et al.*, 2008]. Influence maximization is to select a subset of users in social networks such that they can influence the most number of users in expectation, which often appears in advertisement marketing. Sensor placement is to select a few places from all candidate ones to install sensors such that the remaining uncertainty of the environment is mostly reduced, which often appears in fire detection and water contamination detection.

Note that the problem in Definition 1 assumes a fixed budget B. However, in real-world scenarios, the resources often change over time, and thus the budget B may be not fixed, but change dynamically. For examples, in the application of influence maximization, a company may increase or decrease his investment according to the market situation; in the application of sensor placement, some installed sensors may fail due to aging, or some new sensors may be purchased for placement. In both cases, the budget B will change. In this paper, we focus on the subset selection problem with dynamic cost constraints, presented as follows.

Definition 2 (Subset Selection with Dynamic Cost Constraints)**.** *Given a monotone objective function $f : 2^V \to \mathbb{R}^+$, a monotone cost function $c : 2^V \to \mathbb{R}^+$ and a sequence of changes on the budget B, to find a subset optimizing Eq. (3) for each new B.*

Note that after each change of the budget B, we can view the problem as a static problem with the new budget, and run any algorithm from scratch, which, however, may lead to a long running time and a significantly different solution. As in [Bossek *et al.*, 2019; Doskoč *et al.*, 2020; Assimi *et al.*, 2020; Do and Neumann, 2021], the main focus of this paper is the ability of algorithms adapting to the changes of B. That is, we concern the running time of an algorithm until regaining the ϕ-approximation ratio w.r.t. the new budget B', when starting from the solutions having a desired approximation ratio of ϕ w.r.t. the old budget B.

3 Previous Algorithms

In this section, we introduce three state-of-the-art algorithms for the subset selection problem with cost constraints, and show their approximation performances.

3.1 The Generalized Greedy Algorithm

The generalized greedy algorithm [Zhang and Vorobeychik, 2016] iteratively selects one item maximizing the ratio of the marginal gain on f and $\hat{c}$. After examining all items, the algorithm compares the found subset with the best single item, and returns the better one. Let

$$f(\tilde{X}) = \qquad (5)$$
$$\max\left\{ f(X) \,|\, c(X) \leq B \cdot \frac{\alpha_{\hat{c}}(1 + \alpha_c^2(K_B - 1)(1 - \kappa_c))}{\psi(n) K_B} \right\},$$

where α_c and $\alpha_{\hat{c}}$ are the submodularity ratios of the cost function c and its approximation $\hat{c}$ in Eq. (4), respectively, $\kappa_c = 1 - \min_{v \in V : c(\{v\}) > 0} \frac{c(V) - c(V \setminus \{v\})}{c(\{v\})}$ is the total curvature of c, and $K_B = \max\{|X| \mid c(X) \leq B\}$, i.e., the largest size of a subset satisfying the constraint. $\tilde{X}$ is actually an optimal solution of Eq. (3) with a slightly smaller budget constraint, because $\frac{\alpha_{\hat{c}}(1+\alpha_c^2(K_B-1)(1-\kappa_c))}{\psi(n)K_B} \leq 1$, where the inequality holds by $1-\kappa_c \leq 1/\alpha_c$, $0 \leq \alpha_{\hat{c}}, \alpha_c \leq 1$ and $\psi(n) \geq 1$. Theorem 1 shows that the generalized greedy algorithm can obtain the approximation ratio of $(\alpha_f/2)(1-e^{-\alpha_f})$ w.r.t. $f(\tilde{X})$.

Theorem 1. *[Qian* et al.*, 2017] For subset selection with static cost constraints in Definition 1, the generalized greedy algorithm finds a subset $X \subseteq V$ with*

$$f(X) \geq (\alpha_f/2) \cdot (1-e^{-\alpha_f}) \cdot f(\tilde{X}).$$

To handle the dynamic situation where the budget B changes over time, Roostapour *et al.* [2019] introduced a natural adaptive version of the generalized greedy algorithm. When B increases, the algorithm continues to add items greedily; when B decreases, it iteratively deletes one item which minimizes the ratio of the marginal loss on f and $\hat{c}$. However, this adaptive algorithm cannot adapt the solutions well, and the obtained approximation ratio can be arbitrarily bad, as shown in the following theorem.

Theorem 2. *[Roostapour* et al.*, 2019] For subset selection with dynamic cost constraints in Definition 2, there exist instances of increasing B and decreasing B such that the approximation ratios achieved by the adaptive generalized greedy algorithm are $O(1/n)$ and $O(1/\sqrt{n})$, respectively.*

3.2 The POMC Algorithm

Qian *et al.* [2017] proposed an approach based on Pareto Optimization [Friedrich and Neumann, 2015; Qian *et al.*, 2015] for maximizing a Monotone function with a monotone Cost constraint, called POMC, which can use more time to find better solutions. It represents a subset $X \subseteq V$ by a Boolean vector $\boldsymbol{x} \in \{0,1\}^n$, where the i-th bit $x_i = 1$ iff $v_i \in X$. POMC tries to maximize the objective function f and minimize the approximate cost function $\hat{c}$ simultaneously, by reformulating the original problem Eq. (3) as a bi-objective maximization problem

$$\arg\max\nolimits_{\boldsymbol{x}\in\{0,1\}^n} \big(f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x})\big), \tag{6}$$

$$\text{where } f_1(\boldsymbol{x}) = \begin{cases} -\infty, & \hat{c}(\boldsymbol{x}) > B \\ f(\boldsymbol{x}), & \text{otherwise} \end{cases}, \quad f_2(\boldsymbol{x}) = -\hat{c}(\boldsymbol{x}).$$

The solutions violating the constraint (i.e., with $\hat{c}(X) > B$) are excluded by setting their f_1 values to $-\infty$. Note that the idea of bi-objective reformulation has been used to tackle hard problems, e.g., covering [Friedrich *et al.*, 2010] and balancing [Chica *et al.*, 2010] problems.

After the bi-objective transformation, POMC employs a simple multi-objective EA, i.e., GSEMO [Laumanns *et al.*, 2004; Qian *et al.*, 2019], to solve the bi-objective problem. It starts from the empty set 0^n. In each iteration of POMC, a parent solution $\boldsymbol{x}$ is selected from the population P uniformly at random, and used to generate an offspring solution $\boldsymbol{x}'$ by flipping each bit with probability $1/n$; $\boldsymbol{x}'$ is then used to update P by domination-based comparison as presented in Definition 3. After terminated, POMC returns the solution with the largest f value in P.

Definition 3 (Domination)**.** *For two solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$,*

- $\boldsymbol{x}$ weakly dominates $\boldsymbol{x}'$*, denoted as $\boldsymbol{x} \succeq \boldsymbol{x}'$, if $f_1(\boldsymbol{x}) \geq f_1(\boldsymbol{x}') \wedge f_2(\boldsymbol{x}) \geq f_2(\boldsymbol{x}')$;*
- $\boldsymbol{x}$ dominates $\boldsymbol{x}'$*, denoted as $\boldsymbol{x} \succ \boldsymbol{x}'$, if $\boldsymbol{x} \succeq \boldsymbol{x}'$ and either $f_1(\boldsymbol{x}) > f_1(\boldsymbol{x}')$ or $f_2(\boldsymbol{x}) > f_2(\boldsymbol{x}')$;*
- *they are* incomparable *if neither $\boldsymbol{x} \succeq \boldsymbol{x}'$ nor $\boldsymbol{x}' \succeq \boldsymbol{x}$.*

Let $P_{\max}$ denote the largest size of the population P during the running of POMC, and $\delta_{\hat{c}} = \min\{\hat{c}(X \cup \{v\}) - \hat{c}(X) \mid X \subseteq V, v \notin X\}$ denote the minimum increment on $\hat{c}$ by adding a single item. The following two theorems show that POMC can achieve the approximation ratio of $(\alpha_f/2)(1-e^{-\alpha_f})$ for a fixed budget, and can also regain this approximation ratio when the budget increases.

Theorem 3. *[Qian* et al.*, 2017] For subset selection with static cost constraints in Definition 1, POMC using $\mathbb{E}[T] \leq enBP_{\max}/\delta_{\hat{c}}$ finds a subset $X \subseteq V$ with*

$$f(X) \geq (\alpha_f/2) \cdot (1-e^{-\alpha_f}) \cdot f(\tilde{X}).$$

Theorem 4. *[Roostapour* et al.*, 2019] For subset selection with dynamic cost constraints in Definition 2, when B is increased to B', POMC using $T = rn(B'-B)P_{\max}/\delta_{\hat{c}}$ can regain the approximation ratio of $(\alpha_f/2)(1-e^{-\alpha_f})$ with probability $\Omega(1)$, where $r \geq 8e+1$ is a constant.*

However, for either static or dynamic cases, the running time of POMC depends on $P_{\max}$, B (or $B'-B$) and $1/\delta_{\hat{c}}$, all of which may be exponentially large w.r.t. n, resulting in the exponential running time of POMC.

3.3 The EAMC Algorithm

Bian *et al.* [2020] proposed a simple EA for maximizing a Monotone function with a monotone Cost constraint, called EAMC. It tries to maximize a surrogate objective

$$g(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & |\boldsymbol{x}| = 0, \\ f(\boldsymbol{x})/(1-e^{-\alpha_f \hat{c}(\boldsymbol{x})/B}) & |\boldsymbol{x}| \geq 1, \end{cases} \tag{7}$$

which considers both the original objective f and the cost $\hat{c}$, where $|\cdot|$ denotes the number of 1-bits of a vector. EAMC maintains a solution set $\mathtt{bin}(i)$ for each subset size i, which contains at most two solutions, one with the largest g value (denoted as $\boldsymbol{u}^i$) and the other with the largest f value (denoted as $\boldsymbol{v}^i$) generated-so-far. In each iteration, it generates an offspring solution $\boldsymbol{x}'$ as same as POMC. If $\boldsymbol{x}'$ satisfies the constraint, $\mathtt{bin}(|\boldsymbol{x}'|)$ will be updated by comparing $\boldsymbol{x}'$ with $\boldsymbol{u}^{|\boldsymbol{x}'|}$ and $\boldsymbol{v}^{|\boldsymbol{x}'|}$. After terminated, EAMC returns the solution with the largest f value in the population, which must satisfy the constraint.

Theorem 5. *[Bian* et al.*, 2020] For subset selection with static cost constraints in Definition 1, EAMC using $\mathbb{E}[T] \leq 2en^2(n+1)$ finds a subset $X \subseteq V$ with*

$$f(X) \geq (\alpha_f/2) \cdot (1-e^{-\alpha_f}) \cdot f(\tilde{X}).$$

The above theorem shows that for a fixed budget, EAMC can achieve the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio in $O(n^3)$ expected running time. However, in dynamic environments, the budget B can change to B'. Note that the solutions kept in the population are decided by the g function defined in Eq. (7), which relies on B. After B changes, g cannot characterize the new problem well, and thus the solutions may perform bad for B'.

4 The FPOMC Algorithm

In this section, we propose the Fast Pareto Optimization algorithm for maximizing a Monotone function with a monotone Cost constraint, called FPOMC.

FPOMC also tries to solve the reformulated bi-objective maximization problem Eq. (6). It is actually modified from POMC by introducing a greedy selection strategy. For POMC, the population size is unbounded and the selection is uniform. Thus, it is very likely to select a "bad" solution, which may decrease the efficiency of the algorithm. Inspired from the generalized greedy algorithm, we define a function h to estimate the goodness of a solution in selection, and the solution with the largest h value is selected with a high probability. This selection mechanism can guide the search direction efficiently and accelerate the optimization procedure.

To be specific, the population P is divided into subpopulations $P_0, \dots, P_n$, where $P_i = \{\boldsymbol{x} \in P \mid |\boldsymbol{x}| = i\}$ denotes the set of solutions with size i in P. In each iteration, FPOMC first selects a subset size i (which has corresponding solutions in the population) uniformly at random, and then selects a solution having the largest h value from P_i with probability $1/2$. The function $h_{\boldsymbol{z}}(\boldsymbol{x})$ is defined as

$$h_{\boldsymbol{z}}(\boldsymbol{x}) = \begin{cases} \frac{f(\boldsymbol{x})-f(\boldsymbol{z})}{\hat{c}(\boldsymbol{x})-\hat{c}(\boldsymbol{z})} & \hat{c}(\boldsymbol{x}) > \hat{c}(\boldsymbol{z}), \\ (f(\boldsymbol{x}) - f(\boldsymbol{z})) \cdot C + \hat{c}(\boldsymbol{z}) - \hat{c}(\boldsymbol{x}) & \hat{c}(\boldsymbol{x}) \le \hat{c}(\boldsymbol{z}), \end{cases} \tag{8}$$

where $\boldsymbol{x}$ is a solution in P_i, $\boldsymbol{z}$ is a reference point, and C is a large enough number. Intuitively, $h_{\boldsymbol{z}}(\boldsymbol{x})$ measures the goodness of $\boldsymbol{x}$ by the marginal gain on f and $\hat{c}$ w.r.t. a reference point $\boldsymbol{z}$, and FPOMC selects a solution greedily, i.e., having the largest h value. For each subset size i, FPOMC maintains a reference point, denoted by $\boldsymbol{x}^i$, which is updated adaptively during the running of the algorithm.

The procedure of FPOMC is described in Algorithm 1. Starting from the empty set 0^n (line 1), it repeatedly improves the solutions in each subpopulation P_i (lines 2–22). In each iteration, a solution in P is selected (line 3) according to the SELECT subroutine in Algorithm 2. Then, a solution $\boldsymbol{x}'$ is generated by randomly flipping bits of $\boldsymbol{x}$ (line 4), which is used to update the population P_j (line 7) and the reference point $\boldsymbol{x}^j$ (lines 8–13), respectively. Note that if $\boldsymbol{x}^j$ is updated (line 13), a local search (LS) subroutine in Algorithm 3 is employed to quickly exploit the new solution $\boldsymbol{x}^j$ (line 14), and the generated solution $\boldsymbol{y}$ will be used to update P_{j+1} (lines 15–17). After T iterations, the solution with the largest f value in the population P is output (line 23).

The SELECT subroutine in Algorithm 2 selects a nonempty P_i randomly, and returns a solution $\boldsymbol{x} \in S$ with probability $1/2$, where S denotes the set of solutions in P_i such that $h_{\boldsymbol{z}}(\cdot)$ is maximized. Note that if $\boldsymbol{x}^i$ exists and is contained in S, it is preferred than other solutions in S. To determine S, the reference point $\boldsymbol{z}$ is selected to be the existing $\boldsymbol{x}^k$ with the largest k smaller than i. With the other probability of $1/2$, a solution in $P_i \setminus S$ is selected uniformly at random.

The LS subroutine in Algorithm 3 returns a solution $\boldsymbol{y}$ by adding an item into $\boldsymbol{x}$, such that $h_{\boldsymbol{x}}(\boldsymbol{y})$ is maximized. By Eq. (8), $\boldsymbol{y}$ is actually generated by adding an item with the largest ratio of the marginal gain on f and $\hat{c}$ w.r.t. $\boldsymbol{x}$.

Algorithm 1 FPOMC Algorithm

Input: a monotone objective function f, a monotone approximate cost function $\hat{c}$, and a budget B
Parameter: the number T of iterations
Output: a solution $\boldsymbol{x} \in \{0,1\}^n$ with $\hat{c}(\boldsymbol{x}) \le B$
Process:

```
1: Let $x = x^0 = 0^n$, $P = \{x\}$ and $t = 0$;
2: while $t < T$ do
3:     $x = $ SELECT$(P)$;
4:     Generate $x'$ by flipping each bit of $x$ with prob. $1/n$;
5:     Let $j = |x'|$;
6:     if $0 < \hat{c}(x') \le B$ and $\nexists s \in P_j$ such that $s \succ x'$ then
7:         $P_j = (P_j \setminus \{s \in P_j \mid x' \succeq s\}) \cup \{x'\}$;
8:         if $x^j$ doesn't exist then
9:             Let $x^j = x'$
10:        else
11:            Let $z = x^k$, where $k$ is the largest integer such
               that $k < j$ and $x^k$ exists;
12:            if $h_z(x') \ge \max\{h_z(x^j), h_z(\mathrm{LS}(z))\}$ then
13:                $x^j = x'$;
14:                $y = \mathrm{LS}(x^j)$;
15:                if $\hat{c}(y) \le B$ and $\nexists s \in P_{j+1}$ such that $s \succ y$ then
16:                    $P_{j+1} = (P_{j+1} \setminus \{s \in P_{j+1} \mid y \succeq s\}) \cup \{y\}$
17:                end if
18:            end if
19:        end if
20:    end if
21:    $t = t + 1$
22: end while
23: return $\arg\max_{x \in P} f(x)$
```

5 Theoretical Analysis

In this section, we prove the general approximation bound of FPOMC in Theorem 6, implying that FPOMC can achieve the best known polynomial-time approximation guarantee, i.e., $(\alpha_f/2)(1-e^{-\alpha_f})$. We also prove that when the budget B decreases, FPOMC has already achieved the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio (Theorem 7); when B increases to B', FPOMC can regain the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio in at most $O(nK_{B'}(K_{B'} - K_B))$ expected running time (Theorem 8), where K_B and $K_{B'}$ denote the largest size of a subset satisfying $c(X) \le B$ and $c(X) \le B'$, respectively.

The proof of Theorem 6 relies on Lemma 1, which intuitively means that for any subset, the inclusion of a specific item can improve f by at least a quantity proportional to the current distance to the optimum. As in [Zhang and Vorobeychik, 2016; Qian *et al.*, 2017], we assume that $\forall v \in V$: $\hat{c}(\{v\}) \le B$ and $\delta_{\hat{c}}$ defined in Section 3.2 is larger than 0.

Lemma 1. *[Qian et al., 2017] For any $X \subseteq V$, let $v^* \in \arg\max_{v \notin X} \frac{f(X \cup \{v\}) - f(X)}{\hat{c}(X \cup \{v\}) - \hat{c}(X)}$. It holds that $f(X \cup \{v^*\}) - f(X) \ge \alpha_f \frac{\hat{c}(X \cup \{v^*\}) - \hat{c}(X)}{B} \cdot (f(\tilde{X}) - f(X))$.*

Theorem 6. *For the problem in Definition 1, FPOMC using $\mathbb{E}[T] = O(n^2 K_B)$ finds a subset $X \subseteq V$ with*

$$f(X) \ge (\alpha_f/2) \cdot (1 - e^{-\alpha_f}) \cdot f(\tilde{X}),$$

where $f(\tilde{X})$ is defined in Eq. (5).

Algorithm 2 SELECT(P): Subroutine of FPOMC

```
Input: the population $P$
Output: a solution in $P$ for mutation
Process:
1: Choose a nonempty $P_i$ uniformly at random;
2: Let $z = x^k$, where $k$ is the largest integer such that $k < i$
   and $x^k$ exists;
3: Let $S = \{x \mid x \in \arg\max_{x \in P_i} h_z(x)\}$;
4: if $x^i \in S$ then
5:     Let $s = x^i$
6: else
7:     Let $s$ = a solution uniformly selected from $S$
8: end if
9: Let $x = s$ or a solution uniformly selected from $P_i \setminus S$,
   with equal probability, i.e., $1/2$
10: return $x$
```

Algorithm 3 LS(x): Subroutine of FPOMC

```
Input: the solution $x$
Output: a solution with size $|x| + 1$
Process:
1: Let $y = x$;
2: for $i = 1, 2, \ldots, n$ do
3:     if $x_i = 0$ then
4:         Let $s = x$, and set its $i$-th bit to 1, i.e., $s_i = 1$;
5:         if $h_x(s) \geq h_x(y)$ then
6:             $y = s$
7:         end if
8:     end if
9: end for
10: return $y$
```

Proof. The theorem is proved by analyzing the increase of $J_{\max}$, which is defined as the maximal J such that $\forall 1 \leq i \leq J$, the following two conditions hold:

C1: $\exists \boldsymbol{x}^{i-1}$ such that $D(\boldsymbol{x}^{i-1})$ holds;

C2: $\exists \boldsymbol{x} \in P_i$ such that $h_{\boldsymbol{x}^{i-1}}(\boldsymbol{x}) \geq h_{\boldsymbol{x}^{i-1}}(\boldsymbol{x}^{i-1} \cup \{v_*^{i-1}\})$,

where $D(\boldsymbol{x})$ denotes that $f(\boldsymbol{x}) \geq \left(1 - e^{-\alpha_f \hat{c}(\boldsymbol{x})/B}\right) \cdot f(\tilde{X})$, and $v_*^{i-1} \in \arg\max_{v \in V} h_{\boldsymbol{x}^{i-1}}(\boldsymbol{x}^{i-1} \cup \{v\})$.

We first show that $J_{\max} \geq 1$ after at most $en(K_B + 1)$ expected number of iterations. By lines 9 and 13 of Algorithm 1, $\boldsymbol{x}^0$ can only be replaced by a solution with size 0, thus $\boldsymbol{x}^0$ is always 0^n. Then we have $f(\boldsymbol{x}^0) = \left(1 - e^{-\alpha_f \hat{c}(0^n)/B}\right) \cdot f(\tilde{X}) = 0$, which implies that condition C1 holds for $i = 1$. By the update procedure of FPOMC, 0^n will always be kept in P. In each iteration, 0^n will be selected for mutation with probability at least $1/(K_B + 1)$ by line 3 of Algorithm 1. Flipping a specific 0-bit of 0^n (i.e., adding a specific item) can generate a new solution $\boldsymbol{x}'$ with $|\boldsymbol{x}'| = 1$ such that $h_{\boldsymbol{x}^0}(\boldsymbol{x}') = h_{\boldsymbol{x}^0}(\boldsymbol{x}^0 \cup \{v_*^0\})$. Thus, in each iteration, $\boldsymbol{x}'$ can be generated with probability at least $\frac{1}{K_B+1} \cdot \frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq \frac{1}{en(K_B+1)}$, implying that it needs at most $en(K_B + 1)$ expected number of iterations to generate $\boldsymbol{x}'$. If $\boldsymbol{x}'$ is added into P, condition C2 holds for $i = 1$; otherwise, there must exist a solution $\boldsymbol{s} \in P_1$ such that $f(\boldsymbol{s}) \geq f(\boldsymbol{x}')$ and $\hat{c}(\boldsymbol{s}) \leq \hat{c}(\boldsymbol{x}')$, implying that $h_{\boldsymbol{x}^0}(\boldsymbol{s}) \geq h_{\boldsymbol{x}^0}(\boldsymbol{x}') = h_{\boldsymbol{x}^0}(\boldsymbol{x}^0 \cup \{v_*^0\})$, and thus condition C2 has already hold. Thus, after $en(K_B + 1)$ expected number of iterations, $J_{\max} \geq 1$.

Assume that currently $J_{\max} = J$, we now show that $J_{\max}$ will not decrease. Assume that in some iteration t, a solution $\boldsymbol{x}'$ with $|\boldsymbol{x}'| = j$, $1 \leq j \leq J - 1$, is generated and added into P_j in line 7. We need to show that condition C1 holds for $j + 1$, and condition C2 holds for j and $j + 1$.

First we consider condition C1, i.e., $D(\boldsymbol{x}^j)$ holds. If $\boldsymbol{x}^j$ is not replaced by $\boldsymbol{x}'$ in line 13, C1 trivially holds. Thus, we only need to consider that $\boldsymbol{x}^j$ is replaced. By line 12, we have $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}') \geq h_{\boldsymbol{x}^{j-1}}(\mathrm{LS}(\boldsymbol{x}^{j-1}))$. Note that $\boldsymbol{x}^j$ is replaced by $\boldsymbol{x}'$, we have

$$h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}^j) \geq h_{\boldsymbol{x}^{j-1}}(\mathrm{LS}(\boldsymbol{x}^{j-1})) = h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}). \tag{9}$$

Note that $\hat{c}(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}) > \hat{c}(\boldsymbol{x}^{j-1})$, thus

$$h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}) = \frac{f(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}) - f(\boldsymbol{x}^{j-1})}{\hat{c}(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}) - \hat{c}(\boldsymbol{x}^{j-1})} \geq 0, \tag{10}$$

implying $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}^j) \geq 0$ by Eq. (9). Next, we consider two cases for $\hat{c}(\boldsymbol{x}^j) - \hat{c}(\boldsymbol{x}^{j-1})$.

(1) $\hat{c}(\boldsymbol{x}^j) - \hat{c}(\boldsymbol{x}^{j-1}) \leq 0$, then by Eq. (8), $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}^j) = (f(\boldsymbol{x}^j) - f(\boldsymbol{x}^{j-1})) \cdot C + \hat{c}(\boldsymbol{x}^{j-1}) - \hat{c}(\boldsymbol{x}^j) \geq 0$. Because C is large enough, we have $f(\boldsymbol{x}^j) \geq f(\boldsymbol{x}^{j-1})$. Thus,

$$\begin{aligned} f(\boldsymbol{x}^j) \geq f(\boldsymbol{x}^{j-1}) &\geq \left(1 - e^{-\alpha_f \hat{c}(\boldsymbol{x}^{j-1})/B}\right) \cdot f(\tilde{X}) \\ &\geq \left(1 - e^{-\alpha_f \hat{c}(\boldsymbol{x}^j)/B}\right) \cdot f(\tilde{X}). \end{aligned}$$

(2) $\hat{c}(\boldsymbol{x}^j) - \hat{c}(\boldsymbol{x}^{j-1}) > 0$, then

$$\begin{aligned} \frac{f(\boldsymbol{x}^j) - f(\boldsymbol{x}^{j-1})}{\hat{c}(\boldsymbol{x}^j) - \hat{c}(\boldsymbol{x}^{j-1})} &\geq \frac{f(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}) - f(\boldsymbol{x}^{j-1})}{\hat{c}(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\}) - \hat{c}(\boldsymbol{x}^{j-1})} \\ &\geq \frac{\alpha_f}{B} \cdot \left(f(\tilde{X}) - f\left(\boldsymbol{x}^{j-1}\right)\right), \end{aligned} \tag{11}$$

where the first inequality is by Eqs. (8) and (9), and the second inequality is by Lemma 1. Thus,

$$\begin{aligned} f(\boldsymbol{x}^j) &\geq \alpha_f \frac{\hat{c}\left(\boldsymbol{x}^j\right) - \hat{c}(\boldsymbol{x}^{j-1})}{B} \cdot f(\tilde{X}) \\ &+ \left(1 - \alpha_f \frac{\hat{c}(\boldsymbol{x}^j) - \hat{c}(\boldsymbol{x}^{j-1})}{B}\right)(1 - e^{-\alpha_f \hat{c}(\boldsymbol{x}^{j-1})/B}) f(\tilde{X}) \\ &\geq \left(1 - e^{-\alpha_f \hat{c}\left(\boldsymbol{x}^j\right)/B}\right) \cdot f(\tilde{X}). \end{aligned}$$

where the first inequality holds by Eq. (11) and the definition of $J_{\max}$, and the second inequality holds by $\forall r \in \mathbb{R} : 1 - r \leq e^{-r}$. Combining cases (1) and (2), $D(\boldsymbol{x}^j)$ holds, i.e., condition C1 holds for $j + 1$.

Now we consider condition C2 for j. Before $\boldsymbol{x}'$ is added into P_j, there must exist a solution $\boldsymbol{s} \in P_j$ such that $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{s}) \geq h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}^{j-1} \cup \{v_*^{j-1}\})$ according to the definition of $J_{\max}$. If $\boldsymbol{s}$ is not deleted, condition C2 trivially holds for j. Thus, we only need to consider that $\boldsymbol{s}$ is deleted from P_j in line 7, implying that $f(\boldsymbol{x}') \geq f(\boldsymbol{s})$ and $\hat{c}(\boldsymbol{x}') \leq \hat{c}(\boldsymbol{s})$. By Eq. (10), we have $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{s}) \geq 0$, implying $f(\boldsymbol{s}) \geq f(\boldsymbol{x}^{j-1})$

according to Eq. (8). By classification on $\hat{c}(\boldsymbol{x}^{j-1})$, we will show that

$$h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}') \geq h_{\boldsymbol{x}^{j-1}}(\boldsymbol{s}). \qquad (12)$$

(1) $\hat{c}(\boldsymbol{x}^{j-1}) < \hat{c}(\boldsymbol{x}') \leq \hat{c}(\boldsymbol{s})$. Then, by Eq. (8), $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}') = \frac{f(\boldsymbol{x}')-f(\boldsymbol{x}^{j-1})}{\hat{c}(\boldsymbol{x}')-\hat{c}(\boldsymbol{x}^{j-1})} \geq \frac{f(\boldsymbol{s})-f(\boldsymbol{x}^{j-1})}{\hat{c}(\boldsymbol{s})-\hat{c}(\boldsymbol{x}^{j-1})} = h_{\boldsymbol{x}^{j-1}}(\boldsymbol{s})$.
(2) $\hat{c}(\boldsymbol{x}') \leq \hat{c}(\boldsymbol{x}^{j-1}) < \hat{c}(\boldsymbol{s})$. Then, by Eq. (8), $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}') = (f(\boldsymbol{x}') - f(\boldsymbol{x}^{j-1})) \cdot C + \hat{c}(\boldsymbol{x}^{j-1}) - \hat{c}(\boldsymbol{x}') > \frac{f(\boldsymbol{s})-f(\boldsymbol{x}^{j-1})}{\hat{c}(\boldsymbol{s})-\hat{c}(\boldsymbol{x}^{j-1})} = h_{\boldsymbol{x}^{j-1}}(\boldsymbol{s})$, where the inequality holds for large enough C.
(3) $\hat{c}(\boldsymbol{x}') \leq \hat{c}(\boldsymbol{s}) \leq \hat{c}(\boldsymbol{x}^{j-1})$. Then, by Eq. (8), $h_{\boldsymbol{x}^{j-1}}(\boldsymbol{x}') = (f(\boldsymbol{x}')-f(\boldsymbol{x}^{j-1}))\cdot C+\hat{c}(\boldsymbol{x}^{j-1})-\hat{c}(\boldsymbol{x}') \geq (f(\boldsymbol{s})-f(\boldsymbol{x}^{j-1}))\cdot C + \hat{c}(\boldsymbol{x}^{j-1}) - \hat{c}(\boldsymbol{s}) = h_{\boldsymbol{x}^{j-1}}(\boldsymbol{s})$.
Combining the three cases, we have proved Eq. (12), implying that condition C2 holds for j.

If $\boldsymbol{x}^j$ is updated by $\boldsymbol{x}'$ in line 13, the LS subroutine will generate a solution $\boldsymbol{y}$ with $|\boldsymbol{y}| = j + 1$ such that $h_{\boldsymbol{x}^j}(\boldsymbol{y}) = h_{\boldsymbol{x}^j}(\boldsymbol{x}^j \cup \{v_*^j\})$. If $\boldsymbol{y}$ is added into P_{j+1}, condition C2 holds for $j+1$; otherwise, there must exist a solution $\boldsymbol{s} \in P_{j+1}$ that dominates $\boldsymbol{y}$. Then, similar to the analysis of Eq. (12), we have $h_{\boldsymbol{x}^j}(\boldsymbol{s}) \geq h_{\boldsymbol{x}^j}(\boldsymbol{y}) = h_{\boldsymbol{x}^j}(\boldsymbol{x}^j \cup \{v_*^j\})$. Thus, condition C2 also holds for $j + 1$.

By far, we have shown that $J_{\max}$ will not decrease when the newly generated solution $\boldsymbol{x}'$ satisfies $1 \leq |\boldsymbol{x}'| = j \leq J - 1$. When $j = J$, we only need to consider condition C2 for j, and the proof is similar to the analysis of Eq. (12). Thus, $J_{\max}$ will always not decrease.

Next we consider the increase of $J_{\max}$. Let $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in P_J} h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x})$. By the definition of $J_{\max}$, we have $h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x}^*) \geq h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x}^{J-1}\cup\{v_*^{J-1}\})$. We consider two cases:

(1) $\boldsymbol{x}^J$ exists and $h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x}^J) = h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x}^*)$, then similar to the analysis after Eq. (9), $D(\boldsymbol{x}^J)$ holds, i.e., condition C1 holds for $J + 1$. In each iteration, $\boldsymbol{x}^J$ will be selected with probability at least $1/(2(K_B + 1))$ by line 3. The probability that no bits of $\boldsymbol{x}^J$ are flipped in line 4 is $(1-1/n)^n \geq (1-1/n) \cdot 1/e$, and the generated solution will be added into P_J in line 7 and used to update $\boldsymbol{x}^J$ in line 13. Then, the LS subroutine will generate a solution $\boldsymbol{y}$ such that $h_{\boldsymbol{x}^J}(\boldsymbol{y}) = h_{\boldsymbol{x}^J}(\boldsymbol{x}^J \cup \{v_*^J\})$.
If $\hat{c}(\boldsymbol{y}) \leq B$, $\boldsymbol{y}$ will be used to update P_{J+1} in line 16. If $\boldsymbol{y}$ is not added into P_{J+1}, there must exist a solution $\boldsymbol{s} \in P_{J+1}$ that dominates $\boldsymbol{y}$. Similar to the analysis of Eq. (12), we have $h_{\boldsymbol{x}^J}(\boldsymbol{s}) \geq h_{\boldsymbol{x}^J}(\boldsymbol{y}) = h_{\boldsymbol{x}^J}(\boldsymbol{x}^J \cup \{v_*^J\})$, implying that condition C2 holds for $J + 1$. If $\boldsymbol{y}$ is added into P_{J+1}, condition C2 also holds for $J + 1$. Thus, $J_{\max}$ can increase by 1.
If $\hat{c}(\boldsymbol{y}) > B$, similar to the analysis after Eq. (9), $D(\boldsymbol{y})$ holds, implying $f(\boldsymbol{y}) \geq (1 - e^{-\alpha_f}) \cdot f(\tilde{X})$. Let $\boldsymbol{u} \in \arg\max_{v\in V:\hat{c}(\{v\})\leq B} f(\{v\})$. We have

$$\begin{aligned} f(\boldsymbol{y}) = f(\boldsymbol{x}^J) + (f(\boldsymbol{y}) - f(\boldsymbol{x}^J)) &\leq f(\boldsymbol{x}^J) + f(\boldsymbol{y} \setminus \boldsymbol{x}^J)/\alpha_f \\ &\leq f(\boldsymbol{x}^J) + f(\boldsymbol{u})/\alpha_f \leq (f(\boldsymbol{x}^J) + f(\boldsymbol{u}))/\alpha_f, \end{aligned}$$

where the first inequality holds by Eq. (2), and the last holds by $\alpha_f \in [0, 1]$. In each iteration, $\boldsymbol{u}$ can be generated by selecting 0^n and flipping a specific 0-bit, occurring with probability at least $1/(en(K_B + 1))$. Thus, $\boldsymbol{u}$ can be generated in at most $en(K_B + 1)$ expected number of iterations. According to the updating procedure of P_1 in lines 7 and 16, we know that once $\boldsymbol{u}$ is generated, P will always contain a solution $\boldsymbol{u}' \in P_1$ with $f(\boldsymbol{u}') \geq f(\boldsymbol{u})$. Thus, FPOMC finds a solution with the f value at least $\max\{f(\boldsymbol{x}^J), f(\boldsymbol{u})\} \geq (\alpha_f/2)f(\boldsymbol{y}) \geq (\alpha_f/2)(1 - e^{-\alpha_f}) \cdot f(\tilde{X})$. i.e., the desired approximate solution has been found.

(2) $\boldsymbol{x}^J$ exists and $h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x}^J) < h_{\boldsymbol{x}^{J-1}}(\boldsymbol{x}^*)$, or $\boldsymbol{x}^J$ doesn't exist. The proof is similar to that of case (1), and the only difference is that $\boldsymbol{x}^*$ instead of $\boldsymbol{x}^J$ is selected for mutation.

Combining the two cases, $J_{\max}$ can increase by 1, i.e., conditions C1 and C2 hold for $J + 1$, in $O(K_B)$ iterations, or it has found a solution $\boldsymbol{x}$ with $f(\boldsymbol{x}) \geq (\alpha_f/2)(1-e^{-\alpha_f})\cdot f(\tilde{X})$.

Finally, we examine the total expected number of iterations. Note that $J_{\max}$ can increase to at most K_B, because a solution with size larger than K_B will violate the constraint. To make $J_{\max} \geq 1$, the expected number of iterations is at most $en(K_B + 1)$; to increase $J_{\max}$ from 1 to K_B, the expected number of iterations is at most $O(K_B) \cdot K_B = O(K_B^2)$; to generate $\boldsymbol{u}$, the expected number of iterations is at most $en(K_B + 1)$. Thus, the total expected number of iterations is $O(nK_B)$. Note that in each iteration, FPOMC needs to evaluate the objective value at most $2n + 1$ times (i.e., the cost of evaluating $\boldsymbol{x}'$ and two LS subroutines), implying that the total expected running time is $O(n^2K_B)$. □

Theorem 7. *For the problem in Definition 2, assume FPOMC achieves a $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio, when B decreases to B', FPOMC has already achieved the $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio for the new problem.*

Proof. The result can be derived from the proof of Theorem 6, because B can be replaced by B' directly. □

Theorem 8. *For the problem in Definition 2, assume FPOMC achieves a $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio, when B increases to B', FPOMC using $\mathbb{E}[T] = O(nK_{B'}(K_{B'} - K_B))$ can regain the $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio.*

Proof. Note that FPOMC has run $O(nK_B)$ expected number of iterations, to achieve the $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation ratio for the problem with budget B. For the problem with budget B', the B in the proof of Theorem 6 can be replaced by B' directly, and after $O(nK_B)$ expected number of iterations, $J_{\max}$ has increased to K_B. To achieve the desired approximation guarantee, it is sufficient to continue increasing $J_{\max}$ to $K_{B'}$, whose expected number of iterations is $O(K_{B'}(K_{B'} - K_B))$. Thus, the expected running time to regain the $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation ratio is $O(nK_{B'}(K_{B'} - K_B))$. □

6 Conclusion

This paper proposes a new algorithm FPOMC for subset selection with dynamic cost constraints. When the budget on the constraint changes dynamically, FPOMC can maintain the best known approximation ratio of $(\alpha_f/2)(1 - e^{-\alpha_f})$ efficiently, while previous algorithms either achieve arbitrarily bad approximation ratios or require a long running time to maintain this approximation ratio. FPOMC has shown its superiority theoretically, and it is expected to study the empirical performance of FPOMC in the future.

References

[Assimi *et al.*, 2020] H. Assimi, O. Harper, Y. Xie, A. Neumann, and F. Neumann. Evolutionary bi-objective optimization for the dynamic chance-constrained knapsack problem based on tail bound objectives. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, pages 307–314, Santiago de Compostela, Spain, 2020.

[Bian *et al.*, 2020] C. Bian, C. Feng, C. Qian, and Y. Yu. An efficient evolutionary algorithm for subset selection with general cost constraints. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 3267–3274, New York, NY, 2020.

[Bossek *et al.*, 2019] J. Bossek, F. Neumann, P. Peng, and D. Sudholt. Runtime analysis of randomized search heuristics for dynamic graph coloring. In *Proceedings of the 21st ACM Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1443–1451, Prague, Czech Republic, 2019.

[Chica *et al.*, 2010] M. Chica, Ó. Cordón, S. Damas, and J. Bautista. Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. *Information Sciences*, 180(18):3465–3487, 2010.

[Das and Kempe, 2018] A. Das and D. Kempe. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *Journal of Machine Learning Research*, 19:1–34, 2018.

[Do and Neumann, 2021] A. V. Do and F. Neumann. Pareto optimization for subset selection with dynamic partition matroid constraints. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, Virtual, 2021.

[Doskoč *et al.*, 2020] V. Doskoč, T. Friedrich, A. Göbel, F. Neumann, A. Neumann, and F. Quinzan. Non-monotone submodular maximization with multiple knapsacks in static and dynamic settings. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI)*, pages 435–442, Santiago de Compostela, Spain, 2020.

[Feng *et al.*, 2019] C. Feng, C. Qian, and K. Tang. Unsupervised feature selection by Pareto optimization. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 3534–3541, Honolulu, HI, 2019.

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

[Friedrich *et al.*, 2010] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[Harshaw *et al.*, 2019] C. Harshaw, M. Feldman, J. Ward, and A. Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2634–2643, Long Beach, CA, 2019.

[Kempe *et al.*, 2003] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, Washington, DC, 2003.

[Khuller *et al.*, 1999] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[Krause *et al.*, 2008] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[Laumanns *et al.*, 2004] M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182, 2004.

[Lin and Bilmes, 2011] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 510–520, Portland, OR, 2011.

[Nemhauser *et al.*, 1978] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming*, 14(1):265–294, 1978.

[Qian *et al.*, 2015] C. Qian, Y. Yu, and Z.-H. Zhou. Subset selection by Pareto optimization. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 1765–1773, Montreal, Canada, 2015.

[Qian *et al.*, 2017] C. Qian, J.-C. Shi, Y. Yu, and K. Tang. On subset selection with general cost constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2613–2619, Melbourne, Australia, 2017.

[Qian *et al.*, 2018] Chao Qian, Yang Yu, and Ke Tang. Approximation guarantees of stochastic greedy algorithms for subset selection. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1478–1484, 2018.

[Qian *et al.*, 2019] C. Qian, Y. Yu, K. Tang, X. Yao, and Z.-H. Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence*, 275:279–294, 2019.

[Roostapour *et al.*, 2019] V. Roostapour, A. Neumann, F. Neumann, and T. Friedrich. Pareto optimization for subset selection with dynamic cost constraints. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 2354–2361, Honolulu, HI, 2019.

[Zhang and Vorobeychik, 2016] H. Zhang and Y. Vorobeychik. Submodular optimization with routing constraints. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pages 819–826, Phoenix, AZ, 2016.