On Subset Selection with General Cost Constraints*

Chao Qian[1], **Jing-Cheng Shi**[2], **Yang Yu**[2], **Ke Tang**[1]
[1]UBRI, School of Computer Science and Technology,
University of Science and Technology of China, Hefei 230027, China
[2]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{chaoqian, ketang}@ustc.edu.cn, {shijc, yuy}@lamda.nju.edu.cn

Abstract

This paper considers the subset selection problem with a monotone objective function and a monotone cost constraint, which relaxes the submodular property of previous studies. We first show that the approximation ratio of the generalized greedy algorithm is $\frac{\alpha}{2}(1-\frac{1}{e^{\alpha}})$ (where α is the submodularity ratio); and then propose POMC, an anytime randomized iterative approach that can utilize more time to find better solutions than the generalized greedy algorithm. We show that POMC can obtain the same general approximation guarantee as the generalized greedy algorithm, but can achieve better solutions in cases and applications.

1 Introduction

The problem of selecting a k-element subset that maximizes a monotone submodular objective f lays in the core of many applications, which has been addressed with gradually relaxed assumptions on the constraint.

With cardinality constraints: The subset selection problem with a cardinality constraint $|X| \leq k$ was studied early. It was shown to be NP-hard. The greedy algorithm, which iteratively selects one element with the largest marginal gain, can achieve a $(1-\frac{1}{e})$-approximation guarantee [Nemhauser *et al.*, 1978]. This approximation ratio is optimal in general [Nemhauser and Wolsey, 1978].

With linear cost constraints: After that, the problem with a linear cost constraint $c(X) \leq B$ (where c is a linear function) attracted more attentions. The original greedy algorithm meets some trouble: it has an unbounded approximation ratio [Khuller *et al.*, 1999]. The generalized greedy algorithm was then developed to achieve a $\frac{1}{2}(1-\frac{1}{e})$-approximation guarantee [Krause and Guestrin, 2005], which was further improved to $(1-\frac{1}{\sqrt{e}})$ [Lin and Bilmes, 2010]. The generalized greedy algorithm iteratively selects the element with the largest ratio of the marginal gain on f and c, and finally outputs the better of the found subset and the best single element. By partial enumerations, the generalized greedy algorithm can even achieve a $(1-\frac{1}{e})$-approximation ratio, but with much more computation time [Krause and Guestrin, 2005].

With monotone submodular cost constraints: Iyer and Bilmes [2013] later considered the problem with a monotone submodular cost constraint. By choosing appropriate surrogate functions for f and c to optimize over, several algorithms with bounded approximation guarantees were proposed. However, in some real applications such as mobile robotic sensing and door-to-door marketing, the cost function can be non-submodular [Herer, 1999], which appeals for more general optimization.

With monotone cost constraints: Zhang and Vorobeychik [2016] analyzed the case that the cost function c is only monotone. By introducing the concept of submodularity ratio, they proved that the generalized greedy algorithm achieves a $\frac{1}{2}(1-\frac{1}{e})$-approximation ratio, comparing with the optimal solution with a slightly relaxed budget constraint.

Most of the above-mentioned previous studies considered monotone submodular objective functions. Meanwhile, some applications such as sparse regression and dictionary selection involve non-submodular objective functions [Das and Kempe, 2011]. This paper considers the problem of maximizing **monotone functions with monotone cost constraints**, that is, both the objective function f and the cost function c can be non-submodular.

• First, we extend the analytical results in [Zhang and Vorobeychik, 2016], and derive that the generalized greedy algorithm obtains a $\frac{\alpha_f}{2}(1-\frac{1}{e^{\alpha_f}})$-approximation guarantee (**Theorem 1**), where α_f is the submodularity ratio of f.

• The greedy nature of the generalized greedy algorithm results in an efficient fixed time algorithm, but at the same time limits its performance. We then propose a Pareto optimization [Qian *et al.*, 2015; 2016] method, POMC, which is an anytime algorithm that can use more time to find better solutions. POMC first reformulates the original constrained optimization problem as a bi-objective optimization problem that maximizes f and minimizes c simultaneously, then employs a randomized iterative algorithm to solve it, and finally selects the best feasible solution from the maintained set of solutions. We show that POMC can achieve the same general approximation guarantee as the generalized greedy algorithm (**Theorem 2**). Moreover, in a case of the influence maximization application, POMC can escape the local optimum, while the generalized greedy algorithm cannot (**Theorem 3**).

*This work was supported by the NSFC (61375061, 61603367, 61672478, U1605251, U1613216), the JiangsuSF (BK20160066) and the CCF-Tencent Open Research Fund.

• Experimental results on sensor placement and influence maximization with both cardinality and routing constraints exhibit the superior performance of POMC.

2 Preliminaries

The General Problem. Given a finite set $V = \{v_1, \dots, v_n\}$, we study the functions $f : 2^V \to \mathbb{R}$ over subsets of V. Such a set function f is monotone if for any $X \subseteq Y$, $f(X) \leq f(Y)$. Without loss of generality, we assume that monotone functions are normalized, i.e., $f(\emptyset) = 0$. A set function $f : 2^V \to \mathbb{R}$ is submodular [Nemhauser *et al.*, 1978] if for any $X \subseteq Y \subseteq V$ and $v \notin Y$,

$$f(X \cup v) - f(X) \geq f(Y \cup v) - f(Y); \tag{1}$$

or for any $X \subseteq Y \subseteq V$,

$$f(Y) - f(X) \leq \sum\nolimits_{v \in Y \setminus X} \big(f(X \cup v) - f(X)\big). \tag{2}$$

Note that we represent a set $\{v\}$ with a single element as v.

Two concepts will be used in our analysis. The submodularity ratio in Definition 1 characterizes how close a set function is to submodularity. It is easy to see from Eq. (1) that f is submodular iff $\alpha_f = 1$. Note that an alternative definition based on Eq. (2) was also used in [Das and Kempe, 2011]. The curvature in Definition 2 characterizes how close a monotone submodular set function f is to modularity. It is easy to verify that $1 \geq 1 - \kappa_f(X) \geq 1 - \kappa_f \geq 0$. But in this paper, we also use it for a general monotone function f as in [Zhang and Vorobeychik, 2016]. Without the submodular property, it only holds that $1 - \kappa_f(X) \geq \alpha_f(1 - \kappa_f) \geq 0$.

Definition 1 (Submodularity Ratio [Zhang and Vorobeychik, 2016])**.** *The submodularity ratio of a non-negative set function f is defined as* $\alpha_f = \min_{X \subseteq Y, v \notin Y} \frac{f(X \cup v) - f(X)}{f(Y \cup v) - f(Y)}$.

Definition 2 (Curvature [Conforti and Cornuéjols, 1984; Vondrák, 2010; Iyer *et al.*, 2013])**.** *Let f be a monotone submodular set function. The total curvature of f is*

$$\kappa_f = 1 - \min_{v \in V : f(v) > 0} \big(f(V) - f(V \setminus v)\big)/f(v).$$

The curvature with respect to a set $X \subseteq V$ is

$$\kappa_f(X) = 1 - \min_{v \in X : f(v) > 0} \big(f(X) - f(X \setminus v)\big)/f(v).$$

Our studied problem as presented in Definition 3 is to maximize a monotone objective function f subject to an upper limit on a monotone cost function c. Since the exact computation of $c(X)$ (e.g., a shortest walk to visit a subset of vertices on a graph) may be unsolvable in polynomial time, we assume that only an $\psi(n)$-approximation function $\hat{c}(X)$ can be obtained as in [Zhang and Vorobeychik, 2016], where $c(X) \leq \hat{c}(X) \leq \psi(n) \cdot c(X)$. If $\psi(n) = 1$, $\hat{c}(X) = c(X)$.

Definition 3 (The General Problem)**.** *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, the task is as follows:*

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

Sensor Placement. Sensor placement as presented in Definition 4 is to decide where to take measurements such that the entropy of selected locations is maximized, where o_j denotes a random variable representing the observations collected from location v_j by installing a sensor. Note that the entropy $H(\cdot)$ is monotone and submodular. The traditional sensor placement problem [Krause *et al.*, 2008] has an upper limit on the selected number of sensors, that is, $c(X) = |X|$. However, in mobile robotic sensing domains, both the costs of moving between locations and that of making measurements at locations need to be considered. Let a graph $G(V, E)$ characterize the routing network of all the locations, where c_e is the cost of traversing an edge $e \in E$ and c_v is the cost of visiting a node $v \in V$. The cost function is usually defined as $c(X) = c_R(X) + \sum_{v \in X} c_v$ [Zhang and Vorobeychik, 2016], where $c_R(X)$ is the cost of the shortest walk to visit each node in X at least once (i.e, a variant of the TSP problem). Note that $c_R(X)$ is generally non-submodular [Herer, 1999] and cannot be exactly computed in polynomial time. These two kinds of cost constraints will be called cardinality and routing constraints, respectively.

Definition 4 (Sensor Placement)**.** *Given n locations $V = \{v_1, \dots, v_n\}$, a cost function c and a budget B, the task is as follows:* $\arg\max_{X \subseteq V} H(\{o_j \mid v_j \in X\})$ *s.t.* $c(X) \leq B$.

Influence Maximization. Influence maximization is to identify a set of influential users in social networks. Let a directed graph $G(V, E)$ represent a social network, where each node is a user and each edge $(u, v) \in E$ has a probability $p_{u,v}$ representing the strength of influence from user u to v. As presented in Definition 5, the goal is to find a subset X of V such that the expected number of nodes activated by propagating from X is maximized. A fundamental propagation model is Independence Cascade (IC). Starting from a seed set X, it uses a set A_t to record the nodes activated at time t, and at time $t + 1$, each inactive neighbor v of $u \in A_t$ becomes active with probability $p_{u,v}$; this process is repeated until no nodes get activated at some time. The set of nodes activated by propagating from X is denoted as $IC(X)$, which is a random variable. The objective $\mathbb{E}[|IC(X)|]$ is monotone and submodular. The traditional influence maximization problem [Kempe *et al.*, 2003] is with cardinality constraints, i.e., $c(X) = |X|$. However, some special applications are with routing constraints, such as door-to-door marketing needs to consider the traversing time cost to visit households of choice.

Definition 5 (Influence Maximization)**.** *Given a directed graph $G(V, E)$, edge probabilities $p_{u,v}$ ($(u, v) \in E$), a cost function c and a budget B, the task is as follows:*

$$\arg\max\nolimits_{X \subseteq V} \quad \mathbb{E}[|IC(X)|] \quad \textit{s.t.} \quad c(X) \leq B.$$

3 The Generalized Greedy Algorithm

Zhang and Vorobeychik [2016] have recently investigated the problem of maximizing a monotone submodular function with a monotone cost constraint, i.e., the function f in Definition 3 is submodular. They proposed the generalized greedy algorithm, as shown in Algorithm 1. It iteratively selects one element v such that the ratio of the marginal gain on f and $\hat{c}$ by adding v is maximized; the better of the found subset X and the best single element v^* is finally returned.

Algorithm 1 Generalized Greedy Algorithm

```
Input: a monotone objective function $f$, a monotone approximate cost function $\hat{c}$, and a budget $B$
Output: a solution $X \subseteq V$ with $\hat{c}(X) \leq B$
Process:
 1: Let $X = \emptyset$ and $V' = V$.
 2: repeat
 3:     $v^* \in \arg\max_{v \in V'} \frac{f(X \cup v) - f(X)}{\hat{c}(X \cup v) - \hat{c}(X)}$.
 4:     if $\hat{c}(X \cup v^*) \leq B$ then $X = X \cup v^*$ end if
 5:     $V' = V' \setminus \{v^*\}$.
 6: until $V' = \emptyset$
 7: Let $v^* \in \arg\max_{v \in V: \hat{c}(v) \leq B} f(v)$.
 8: return $\arg\max_{S \in \{X, v^*\}} f(S)$
```

Let $K_c = \max\{|X| \mid c(X) \leq B\}$, i.e., the largest size of a feasible solution. Let $\tilde{X}$ be a corresponding solution to

$$\max \left\{ f(X) \mid c(X) \leq \alpha_{\hat{c}} \frac{B(1+\alpha_c^2(K_c-1)(1-\kappa_c))}{\psi(n)K_c} \right\}, \quad (4)$$

i.e., an optimal solution of the original problem Eq. (3) with a slightly smaller budget constraint. The generalized greedy algorithm was proved to achieve a $\frac{1}{2}(1-\frac{1}{e})$-approximation ratio of $f(\tilde{X})$. The key of that proof is Lemma 1, that is, the inclusion of the greedily selected element can improve f by at least a quantity proportional to the current distance to the optimum. Note that in their original proof, $1 - \kappa_c(X) \geq 1 - \kappa_c$ is used, which is, however, not true. For a general monotone cost function $c(X)$, it only holds that $1 - \kappa_c(X) \geq \alpha_c(1 - \kappa_c)$ by Definitions 1 and 2. We have corrected their results by replacing α_c with α_c^2 in Eq. (4).

Lemma 1. *[Zhang and Vorobeychik, 2016] Let X_i be the subset generated after the i-th iteration of Algorithm 1 until the first time the budget constraint is violated. For the problem in Definition 3 with f being submodular, it holds that*

$$f(X_i) - f(X_{i-1}) \geq \frac{\hat{c}(X_i) - \hat{c}(X_{i-1})}{B} \cdot (f(\tilde{X}) - f(X_{i-1})).$$

We extend their results to the general situation, i.e., f is not necessarily submodular. As shown in Theorem 1, the approximation ratio now is $\frac{\alpha_f}{2}(1 - \frac{1}{e^{\alpha_f}})$. The proofs of Lemma 2 and Theorem 1 are similar to that of Lemma 1 and the theorem in [Zhang and Vorobeychik, 2016]. The major difference is that instead of Eq. (1), $f(X \cup v) - f(X) \geq \alpha_f \cdot (f(Y \cup v) - f(Y))$ is used for a general monotone objective function f.

Lemma 2. *Let X_i be the subset generated after the i-th iteration of Algorithm 1 until the first time the budget constraint is violated. For the problem in Definition 3, it holds that*

$$f(X_i) - f(X_{i-1}) \geq \alpha_f \frac{\hat{c}(X_i) - \hat{c}(X_{i-1})}{B} \cdot (f(\tilde{X}) - f(X_{i-1})).$$

Theorem 1. *For the problem in Definition 3, the generalized greedy algorithm finds a subset $X \subseteq V$ with*

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

4 The POMC Approach

The greedy nature of the generalized greedy algorithm may limit its performance. To alleviate the issue of getting trapped in local optima, we propose a new approach based on Pareto Optimization [Qian *et al.*, 2015] for maximizing a Monotone function with a monotone Cost constraint, called POMC.

Algorithm 2 POMC 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 $\hat{c}(\boldsymbol{x}) \leq B$
Process:
 1: Let $\boldsymbol{x} = \{0\}^n$ and $P = \{\boldsymbol{x}\}$.
 2: Let $t = 0$.
 3: while $t < T$ do
 4:     Select $\boldsymbol{x}$ from $P$ uniformly at random.
 5:     Generate $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ with prob. $1/n$.
 6:     if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succ \boldsymbol{x}'$ then
 7:         $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{x}' \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$.
 8:     end if
 9:     $t = t + 1$.
10: end while
11: return $\arg\max_{\boldsymbol{x} \in P: \hat{c}(\boldsymbol{x}) \leq B} f(\boldsymbol{x})$
```

POMC reformulates the original problem Eq. (3) as a bi-objective maximization problem

$$\arg\max_{\boldsymbol{x} \in \{0,1\}^n} \quad \big(f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x})\big),$$

$$\text{where } f_1(\boldsymbol{x}) = \begin{cases} -\infty, & \hat{c}(\boldsymbol{x}) \geq 2B \\ f(\boldsymbol{x}), & \text{otherwise} \end{cases}, \quad f_2(\boldsymbol{x}) = -\hat{c}(\boldsymbol{x}).$$

That is, POMC maximizes the objective function f and minimizes the approximate cost function $\hat{c}$ simultaneously. Setting f_1 to $-\infty$ is to exclude overly infeasible solutions. Note that we use a Boolean vector $\boldsymbol{x} \in \{0,1\}^n$ to represent a subset $X \subseteq V$, where the i-th bit $x_i = 1$ means that $v_i \in X$, and $x_i = 0$ means that $v_i \notin X$. In this paper, we will not distinguish $\boldsymbol{x} \in \{0,1\}^n$ and its corresponding subset X.

In the bi-objective setting, both the two objective values have to be considered for comparing two solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$. $\boldsymbol{x}$ *weakly dominates* $\boldsymbol{x}'$ (i.e., $\boldsymbol{x}$ is *better* than $\boldsymbol{x}'$, 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* $\boldsymbol{x}'$ (i.e., $\boldsymbol{x}$ is *strictly better*, denoted as $\boldsymbol{x} \succ \boldsymbol{x}'$) if $\boldsymbol{x} \succeq \boldsymbol{x}'$ and either $f_1(\boldsymbol{x}) > f_1(\boldsymbol{x}')$ or $f_2(\boldsymbol{x}) > f_2(\boldsymbol{x}')$. But if neither $\boldsymbol{x}$ is better than $\boldsymbol{x}'$ nor $\boldsymbol{x}'$ is better than $\boldsymbol{x}$, they are *incomparable*.

The procedure of POMC is described in Algorithm 2. Starting from the empty set $\{0\}^n$ (line 1), it repeatedly tries to improve the quality of solutions in the archive P (lines 3-10). In each iteration, a new solution $\boldsymbol{x}'$ is generated by randomly flipping bits of an archived solution $\boldsymbol{x}$ selected from the current P (lines 4-5); if $\boldsymbol{x}'$ is not dominated by any previously archived solution (line 6), it will be added into P, and meanwhile those solutions worse than $\boldsymbol{x}'$ will be removed (line 7). Note that P always contains incomparable solutions.

POMC repeats for T iterations. The value of T is a parameter, which could affect the quality of the produced solution. Their relationship will be analyzed in the next section, and we will use the theoretically derived T value in the experiments. After the iterations, the solution having the largest f value and satisfying the budget constraint in P is selected (line 11).

Compared with the generalized greedy algorithm, POMC may escape local optima by two different ways: (1) *backward search*, i.e., flipping one bit value from 1 to 0; (2) *multi-bit search*, i.e., flipping more than one 0-bits to 1-bits simultaneously. This advantage of POMC over the generalized greedy algorithm will be theoretically shown in the next section.

5 Approximation Guarantee of POMC

We first prove the general approximation bound of POMC in Theorem 2, where $\mathbb{E}[T]$ denotes the expected number of iterations. Let $P_{\max}$ denote the largest size of P during the run of POMC, and let $\delta_{\hat{c}} = \min\{\hat{c}(X \cup v) - \hat{c}(X) \mid X \subseteq V, v \notin X\}$. Note that we assume that $\delta_{\hat{c}} > 0$. The proof relies on Lemma 3, which can be directly derived from Lemma 2. This is because the proof of Lemma 2 does not depend on X_{i-1}, but only requires that the element added into X_{i-1} has the largest ratio of the marginal gain on f and $\hat{c}$.

Lemma 3. *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*

$$f(X \cup v^*) - f(X) \geq \alpha_f \frac{\hat{c}(X \cup v^*) - \hat{c}(X)}{B} \cdot (f(\tilde{X}) - f(X)).$$

Theorem 2. *For the problem in Definition 3, POMC with $\mathbb{E}[T] \leq enBP_{\max}/\delta_{\hat{c}}$ finds a subset $X \subseteq V$ with*

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

Proof. The proof is accomplished by analyzing the increase of a quantity $J_{\max}$, which is defined as $J_{\max} = \max\{j \in [0, B) \mid \exists \boldsymbol{x} \in P, \hat{c}(\boldsymbol{x}) \leq j \wedge f(\boldsymbol{x}) \geq (1 - (1 - \alpha_f \frac{j}{Bk})^k) \cdot f(\tilde{X})$ for some $k\}$.

The initial value of $J_{\max}$ is 0, since POMC starts from $\{0\}^n$. Assume that currently $J_{\max} = i < B$. Let $\boldsymbol{x}$ be the corresponding solution with the value i, i.e., $\hat{c}(\boldsymbol{x}) \leq i$ and

$$f(\boldsymbol{x}) \geq \left(1 - \left(1 - \alpha_f \tfrac{i}{Bk}\right)^k\right) \cdot f(\tilde{X}) \quad \text{for some } k.$$

We first show that $J_{\max}$ cannot decrease. If $\boldsymbol{x}$ is kept in P, $J_{\max}$ obviously will not decrease. If $\boldsymbol{x}$ is deleted from P (line 7 of Algorithm 2), the newly included solution $\boldsymbol{x}'$ must weakly dominate $\boldsymbol{x}$, i.e., $f(\boldsymbol{x}') \geq f(\boldsymbol{x})$ and $\hat{c}(\boldsymbol{x}') \leq \hat{c}(\boldsymbol{x})$, which makes $J_{\max} \geq i$.

We then show that $J_{\max}$ can increase by at least $\delta_{\hat{c}}$ in each iteration with probability at least $\frac{1}{enP_{\max}}$. By Lemma 3, we know that flipping one specific 0-bit of $\boldsymbol{x}$ (i.e., adding a specific element) can generate a new solution $\boldsymbol{x}'$ such that

$$f(\boldsymbol{x}') - f(\boldsymbol{x}) \geq \alpha_f \frac{\hat{c}(\boldsymbol{x}') - \hat{c}(\boldsymbol{x})}{B}(f(\tilde{X}) - f(\boldsymbol{x})).$$

By applying the above two inequalities, we get

$$\begin{aligned} f(\boldsymbol{x}') &\geq \left(1 - \left(1 - \alpha_f \tfrac{i}{Bk}\right)^k \left(1 - \alpha_f \tfrac{\hat{c}(\boldsymbol{x}') - \hat{c}(\boldsymbol{x})}{B}\right)\right) \cdot f(\tilde{X}) \\ &\geq \left(1 - \left(1 - \alpha_f \tfrac{i + \hat{c}(\boldsymbol{x}') - \hat{c}(\boldsymbol{x})}{B(k+1)}\right)^{k+1}\right) \cdot f(\tilde{X}), \end{aligned}$$

where the second inequality is by applying the AM-GM inequality. Since $\hat{c}(\boldsymbol{x}') \leq i + \hat{c}(\boldsymbol{x}') - \hat{c}(\boldsymbol{x})$, $\boldsymbol{x}'$ will be included into P; otherwise, $\boldsymbol{x}'$ must be dominated by one solution in P (line 6 of Algorithm 2), and this implies that $J_{\max}$ has already been larger than i, which contradicts with the assumption $J_{\max} = i$. After including $\boldsymbol{x}'$, $J_{\max} \geq i + \hat{c}(\boldsymbol{x}') - \hat{c}(\boldsymbol{x}) \geq i + \delta_{\hat{c}}$. Let $P_{\max}$ denote the largest size of P during the run of POMC. Thus, $J_{\max}$ can increase by at least $\delta_{\hat{c}}$ in one iteration with probability at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq \frac{1}{enP_{\max}}$, where $\frac{1}{P_{\max}}$ is a lower bound on the probability of selecting $\boldsymbol{x}$ in line 4 of Algorithm 2 due to uniform selection and $\frac{1}{n}(1 - \frac{1}{n})^{n-1}$ is the probability of flipping a specific bit of $\boldsymbol{x}$ and keeping other bits unchanged in line 5. Then, it needs at most $enP_{\max}$ expected number of iterations to increase $J_{\max}$ by at least $\delta_{\hat{c}}$.

Let $v^* \in \arg\max_{v \notin \boldsymbol{x}} \frac{f(\boldsymbol{x} \cup v) - f(\boldsymbol{x})}{\hat{c}(\boldsymbol{x} \cup v) - \hat{c}(\boldsymbol{x})}$. It is easy to see that after at most $\frac{B}{\delta_{\hat{c}}} \cdot enP_{\max}$ expected number of iterations, $J_{\max} + \hat{c}(\boldsymbol{x} \cup v^*) - \hat{c}(\boldsymbol{x}) \geq B$. This implies that there exists one solution $\boldsymbol{x}$ in P satisfying that $\hat{c}(\boldsymbol{x}) \leq J_{\max} < B$ and

$$\begin{aligned} f(\boldsymbol{x} \cup v^*) &\geq \left(1 - \left(1 - \alpha_f \tfrac{J_{\max} + \hat{c}(\boldsymbol{x} \cup v^*) - \hat{c}(\boldsymbol{x})}{Bk}\right)^k\right) \cdot f(\tilde{X}) \\ &\geq \left(1 - \left(1 - \alpha_f \tfrac{B}{Bk}\right)^k\right) \cdot f(\tilde{X}) \geq \left(1 - \tfrac{1}{e^{\alpha_f}}\right) \cdot f(\tilde{X}). \end{aligned}$$

Let $\boldsymbol{y} = \arg\max_{v \in V : \hat{c}(v) \leq B} f(v)$. We then get

$$\begin{aligned} f(\boldsymbol{x} \cup v^*) &= f(\boldsymbol{x}) + (f(\boldsymbol{x} \cup v^*) - f(\boldsymbol{x})) \leq f(\boldsymbol{x}) + f(v^*)/\alpha_f \\ &\leq f(\boldsymbol{x}) + f(\boldsymbol{y})/\alpha_f \leq (f(\boldsymbol{x}) + f(\boldsymbol{y}))/\alpha_f, \end{aligned}$$

where the first inequality is by Definition 1, and the last is by $\alpha_f \in [0, 1]$. Thus, after at most $enBP_{\max}/\delta_{\hat{c}}$ iterations in expectation, P must contain a solution $\boldsymbol{x}$ with $\hat{c}(\boldsymbol{x}) \leq B$ and

$$f(\boldsymbol{x}) + f(\boldsymbol{y}) \geq \alpha_f(1 - 1/e^{\alpha_f}) \cdot f(\tilde{X}).$$

Note that $\{0\}^n$ will always be in P, since it has the smallest $\hat{c}$ value and no other solutions can dominate it. Thus, $\boldsymbol{y}$ can be generated in one iteration by selecting $\{0\}^n$ in line 4 of Algorithm 2 and flipping only the corresponding 0-bit in line 5, whose probability is at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq \frac{1}{enP_{\max}}$. That is, $\boldsymbol{y}$ will be generated in at most $enP_{\max}$ expected iterations. According to the updating procedure of P (lines 6-8), we know that once $\boldsymbol{y}$ is produced, P will always contain a solution $\boldsymbol{z} \succeq \boldsymbol{y}$, i.e., $\hat{c}(\boldsymbol{z}) \leq \hat{c}(\boldsymbol{y}) \leq B$ and $f(\boldsymbol{z}) \geq f(\boldsymbol{y})$.

By line 11 of Algorithm 2, the best solution satisfying the budget constraint will be finally returned. Thus, POMC using $\mathbb{E}[T] \leq enBP_{\max}/\delta_{\hat{c}}$ finds a solution with the f value at least

$$\max\{f(\boldsymbol{x}), f(\boldsymbol{y})\} \geq (\alpha_f/2) \cdot (1 - 1/e^{\alpha_f}) \cdot f(\tilde{X}). \qquad \square$$

The above theorem shows that POMC can obtain the same general approximation ratio as the generalized greedy algorithm. By using an illustrative example of influence maximization with cardinality constraints, we then prove in Theorem 3 that the generalized greedy algorithm will get trapped in local optima, while POMC can find the global optimum. As shown in Example 1, it has a unique global optimal solution $\{v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_{2k-1}\}$. The proof idea is that the generalized greedy algorithm will first select v_k due to the greedy nature and will be misled by it, while POMC can avoid v_k by backward search and multi-bit search, which will be shown in the proof, respectively.

Example 1. *The parameters of influence maximization with cardinality constraints in Definition 5 are set as: the graph $G(V, E)$ is shown in Figure 1 where each edge has a probability 1 and $n = 4k + 5$, and the budget $B = 2k - 2$.*

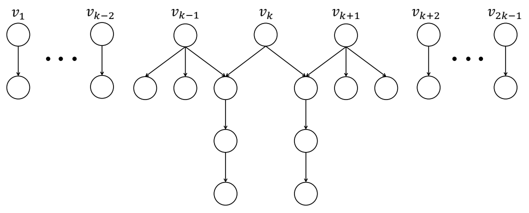


Figure 1: A social network graph.

Lemma 4 (Multiplicative Drift). *[Doerr et al., 2012] Let $S \subseteq \mathbb{R}^+$ be a set of positive numbers with minimum $s_{\min}$. Let $\{X_t\}_{t\in\mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let τ be the random variable that denotes the first time for which $X_t = 0$. If there exists a real number $\delta > 0$ such that $\mathbb{E}[X_t - X_{t+1} \mid X_t = s] \geq \delta s$ holds for all $s \in S$, then for all $s_0 \in S$, we have $\mathbb{E}[\tau \mid X_0 = s_0] \leq \big(1 + \log(s_0/s_{min})\big)/\delta$.*

Theorem 3. *For Example 1, POMC with $\mathbb{E}[T] = O(Bn\log n)$ finds the optimal solution, while the generalized greedy algorithm cannot.*

Proof. Since each edge probability is 1, the objective $f(\boldsymbol{x})$ is just the number of nodes reached from X. It is easy to verify that the solution $\{v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_{2k-1}\}$ with the f value $n-1$ is optimal. For the generalized greedy algorithm, we only need to consider the gain on f since the gain on the cost c is always 1 for cardinality constraints. It first selects v_k, which has the largest f value 7. By the procedure of Algorithm 1, we know that the output solution must contain v_k, which implies that the optimal solution cannot be found.

For the POMC algorithm, the problem is implemented as maximizing $f(\boldsymbol{x})$ and minimizing $|\boldsymbol{x}|$ simultaneously. We then prove that POMC can find the optimal solution $\{v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_{2k-1}\}$ by two different ways.

[Backward search] The idea is that POMC first efficiently finds a solution with the maximum f value n, then reaches the solution $\{v_1, \ldots, v_{2k-1}\}$ by deleting elements v_i with $i \geq 2k$, and finally deleting v_k can produce the optimal solution.

Let $F_t = \max\{f(\boldsymbol{x}) | \boldsymbol{x} \in P\}$ after t iterations of POMC. We first use Lemma 4 to derive the number of iterations (denoted as T_1) until $F_t = n$. Let $X_t = n - F_t$. Then, the variable τ in Lemma 4 is just T_1, because $X_t = 0$ is equivalent to $F_t = n$. Since $\mathbb{E}[X_t - X_{t+1} | X_t] = \mathbb{E}[F_{t+1} - F_t | F_t]$, we only need to analyze the change of F_t. Let $\hat{\boldsymbol{x}}$ be the corresponding solution with $f(\hat{\boldsymbol{x}}) = F_t$. First, F_t will not decrease, since deleting $\hat{\boldsymbol{x}}$ in line 7 of Algorithm 2 implies that the newly included solution $\boldsymbol{x}' \succeq \hat{\boldsymbol{x}}$, i.e, $f(\boldsymbol{x}') \geq f(\hat{\boldsymbol{x}})$. We then show that F_t can increase by flipping only one 0-bit of $\hat{\boldsymbol{x}}$. Consider that $\hat{\boldsymbol{x}}$ is selected in line 4 and only $\hat{x}_i$ is flipped in line 5, whose probability is at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1}$. If $\hat{x}_i = 0$, the newly generated solution $\boldsymbol{x}' = \hat{\boldsymbol{x}} \cup v_i$. By the monotonicity of f, $f(\boldsymbol{x}') = f(\hat{\boldsymbol{x}} \cup v_i) \geq f(\hat{\boldsymbol{x}})$. If the inequality strictly holds, $\boldsymbol{x}'$ now has the largest f value and will be added into P, which leads to $F_{t+1} = f(\hat{\boldsymbol{x}} \cup v_i) > F_t$. If $f(\hat{\boldsymbol{x}} \cup v_i) = f(\hat{\boldsymbol{x}})$, obviously $F_{t+1} = F_t = f(\hat{\boldsymbol{x}} \cup v_i)$. Thus,

$$\mathbb{E}[X_t - X_{t+1} | X_t] = \mathbb{E}[F_{t+1} - F_t | F_t] \geq \sum_{i:\hat{x}_i=0} \frac{f(\hat{\boldsymbol{x}} \cup v_i) - f(\hat{\boldsymbol{x}})}{enP_{\max}}$$
$$= \frac{\sum_{v\in V\setminus\hat{\boldsymbol{x}}} \big(f(\hat{\boldsymbol{x}}\cup v) - f(\hat{\boldsymbol{x}})\big)}{enP_{\max}} \geq \frac{f(V) - f(\hat{\boldsymbol{x}})}{enP_{\max}} = \frac{n - F_t}{enP_{\max}} = \frac{X_t}{enP_{\max}},$$

where the second inequality is by the submodularity of f, i.e., Eq. (2). Note that $X_0 \leq n \wedge s_{\min} = 1$. By Lemma 4, we get $\mathbb{E}[T_1] = \mathbb{E}[\tau \mid X_0] \leq enP_{\max}(1 + \log n)$.

From Example 1, we know that a solution $\boldsymbol{x}$ with $f(\boldsymbol{x}) = n$ must contain $v_1, \ldots, v_{2k-1}$. Let $\boldsymbol{x^i}$ $(0 \leq i \leq n - (2k-1))$ denote a solution with $f(\boldsymbol{x^i}) = n$ and $|\boldsymbol{x^i}| = 2k - 1 + i$. P will always contain exactly one $\boldsymbol{x^i}$, because any solution with the f value smaller than n cannot dominate $\boldsymbol{x^i}$ and the solutions in P are incomparable. By selecting $\boldsymbol{x^i}$ in line 4 and flipping only one of the last i 1-bits, whose probability is at least $\frac{1}{P_{\max}} \frac{i}{n}(1-\frac{1}{n})^{n-1}$, $\boldsymbol{x^{i-1}}$ will be generated. Since $\boldsymbol{x^{i-1}} \succ \boldsymbol{x^i}$, $\boldsymbol{x^i}$ will be replaced by $\boldsymbol{x^{i-1}}$. Let T_2 denote the number of iterations until P contains $\boldsymbol{x^0}$. Thus, we get $\mathbb{E}[T_2] \leq \sum_{i=1}^{n-(2k-1)} \frac{enP_{\max}}{i} \leq enP_{\max}(1 + \log n)$.

After finding $\boldsymbol{x^0} = \{v_1, \ldots, v_{2k-1}\}$, the optimal solution can be generated by selecting $\boldsymbol{x^0}$ in line 4 and flipping only the 1-bit corresponding to v_k in line 5, whose probability is at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1}$. Denote the number of iterations in this phase as T_3. We then have $\mathbb{E}[T_3] \leq enP_{\max}$.

Since P only contains incomparable solutions, each value of one objective can correspond to at most one solution in P. The solutions with $|\boldsymbol{x}| \geq 2B$ have the f value $-\infty$, and must be excluded from P. Thus, $P_{\max} \leq 2B$. By combining the above three phases, we get that the expected number of iterations for POMC finding the optimal solution is

$$\mathbb{E}[T] \leq \mathbb{E}[T_1] + \mathbb{E}[T_2] + \mathbb{E}[T_3] = O(Bn\log n).$$

[Multi-bit search] Let $\boldsymbol{x^i}$ $(2 \leq i \leq 2k-2)$ denote the best solution with $|\boldsymbol{x}| = i$. It is easy to verify that $\boldsymbol{x^i}$ must contain v_{k-1} and v_{k+1}, and inserting a specific element into $\boldsymbol{x^i}$ can generate one $\boldsymbol{x^{i+1}}$. The idea is that flipping the two 0s (corresponding to v_{k-1} and v_{k+1}) of the solution $\{0\}^n$ simultaneously can find $\boldsymbol{x^2}$; then following the path $\boldsymbol{x^2} \to \boldsymbol{x^3} \to \cdots \to \boldsymbol{x^{2k-2}}$ can produce the optimal solution.

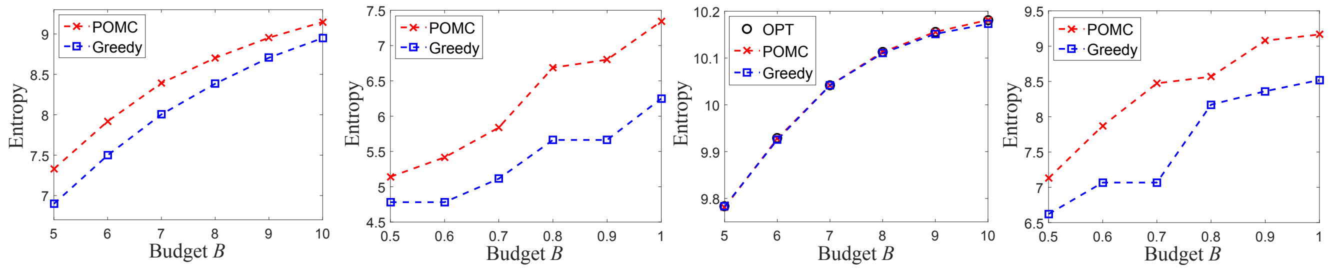
The probability of generating $\boldsymbol{x^2}$ from $\{0\}^n$ is at least $\frac{1}{P_{\max}} \cdot \frac{1}{n^2}(1-\frac{1}{n})^{n-2} \geq \frac{1}{en^2P_{\max}}$. Note that once $\boldsymbol{x^i}$ is generated, it will always be in P, since it cannot be dominated by any other solution. The probability of $\boldsymbol{x^i} \to \boldsymbol{x^{i+1}}$ is at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{enP_{\max}}$, since it is sufficient to select $\boldsymbol{x^i}$ in line 4 and then flip only one specific 0-bit. Thus,

$$\mathbb{E}[T] \leq en^2P_{\max} + (2k-4)\cdot enP_{\max} = O(Bn^2).$$

Taking the minimum of the expected number of iterations for finding the optimal solution by backward search and multi-bit search, the theorem holds. □

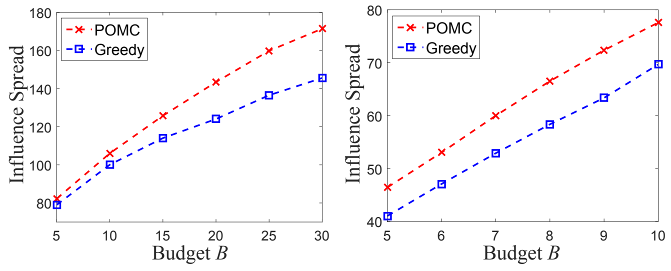
6 Experiments

We empirically compare POMC with the generalized greedy algorithm (denoted as Greedy), the previous best algorithm [Zhang and Vorobeychik, 2016], on sensor placement and influence maximization with both cardinality and routing constraints. Note that we implement the enhancement of Greedy, that is, elements continue to be added in line 7 of Algorithm 1 until the budget constraint is violated. As POMC is a randomized algorithm, we repeat the run 10 times independently and report the average results.



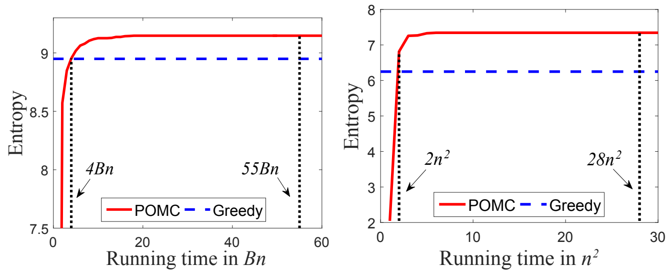
(a) (Berkeley, cardinality) (b) (Berkeley, routing) (c) (Beijing, cardinality) (d) (Beijing, routing)

Figure 2: Sensor placement (entropy: the larger the better).



(a) (Digg, cardinality) (b) (Synthetic, routing)

Figure 3: Influence Maximization (spread: the larger the better).



(a) Berkeley, cardinality, $B = 10$ (b) Berkeley, routing, $B = 1$

Figure 4: Performance v.s. running time of POMC.

Sensor Placement. We use two real-world data sets: one (`http://db.csail.mit.edu/labdata/labdata.html`) is collected from sensors installed at 55 locations of the Intel Berkeley Research lab; the other [Zheng *et al.*, 2013] is air quality data collected from 36 monitoring stations in Beijing. The light (discretized into 5 bins with equal size) and temperature measures are extracted, respectively. The routing network is constructed as a complete graph, where the edge cost corresponds to the physical distance between two locations (normalized to [0,1]) and the node cost is set as 0.1. Note that the entropy is calculated using the observed frequency; the routing cost is approximately computed by the nearest neighbor method [Rosenkrantz *et al.*, 1977].

For cardinality and routing constraints, the budget B is set as $\{5, 6, \ldots, 10\}$ and $\{0.5, 0.6, \ldots, 1\}$, respectively. The results plotted in Figure 2 show that POMC is better than Greedy in most cases, and never worse. Note that on Figure 2(c) (where OPT denotes the optimal solution by exhaustive enumeration), Greedy has already been nearly optimal.

Influence Maximization. For cardinality constraints, we use a real-world data set (`http://www.isi.edu/~lerman/downloads/digg2009.html`) collected from the social news website Digg. It contains two tables, the friendship links between users and the user votes on news stories [Hogg and Lerman, 2012]. After preprocessing, we get a directed graph with 3523 nodes and 90244 edges. We use the method in [Barbieri *et al.*, 2012] to estimate the edge probabilities from the user votes. For routing constraints, we use simulated networks. A social network with 400 nodes is constructed by the well-known Barabasi-Albert (BA) model [Albert and Barabási, 2002], and the edge probability is set as 0.1. A routing network is constructed by the Erdos-Renyi (ER) model [ErdősS and Rényi, 1959], which adds one edge with probability 0.02; the node cost is set as 0.1 and the edge cost is the random Euclidean distance between nodes.

For estimating the influence spread, i.e., the expected number of active nodes, we simulate the diffusion process 1,000 times independently and use the average as an estimation. The results in Figure 3 show that POMC performs better.

Running Time. For the number T of iterations of POMC, we used $enBP_{\max}/\delta_{\hat{c}}$ suggested by Theorem 2. For cardinality constraints, T is $2eB^2n$, since $P_{\max} \leq 2B$ (as in the proof of Theorem 3) and $\delta_{\hat{c}} = 1$; Greedy takes the time in the order of Bn. Since $2eB^2n$ is the theoretical upper bound (i.e., a worst case) for POMC being good, we empirically examine how effective POMC is in practice. By selecting Greedy as the baseline, we plot the curve of the entropy over the running time for POMC on the Berkeley data with $B = 10$, as shown in Figure 4(a). The x-axis is in Bn, the running time of Greedy. We can observe that POMC takes about only 7% (4/55) of the worst-case time to achieve a better performance. This implies that POMC can be efficient in practice.

For routing constraints, we set $P_{\max} = n$: when the number of solutions in P exceeds n, we delete the solution with the smallest ratio of f and $\hat{c}$ except the best feasible one. Since at least a node cost 0.1 is increased, $\delta_{\hat{c}} \geq 0.1$. We thus used $T = 10eBn^2$ for POMC. The time of Greedy is in the order of n^2. From Figure 4(b), we can observe that POMC also quickly reaches a better performance.

7 Conclusion

In this paper, we study the problem of maximizing monotone functions with monotone cost constraints. We first extend the previous analysis to show the approximation ratio of the generalized greedy algorithm. We then propose a new algorithm POMC, and prove that POMC can obtain the same general approximation ratio as the generalized greedy algorithm, but can have a better ability of avoiding local optima. The superior performance of POMC is also empirically verified.

References

[Albert and Barabási, 2002] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[Barbieri *et al.*, 2012] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM'12)*, pages 81–90, Brussels, Belgium, 2012.

[Conforti and Cornuéjols, 1984] M. Conforti and G. Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984.

[Das and Kempe, 2011] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 1057–1064, Bellevue, WA, 2011.

[Doerr *et al.*, 2012] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[ErdősS and Rényi, 1959] P. ErdősS and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[Herer, 1999] Y. Herer. Submodularity and the traveling salesman problem. *European Journal of Operational Research*, 114(3):489–508, 1999.

[Hogg and Lerman, 2012] T. Hogg and K. Lerman. Social dynamics of digg. *EPJ Data Science*, 1(5):1–26, 2012.

[Iyer and Bilmes, 2013] R. Iyer and J. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*, pages 2436–2444, Lake Tahoe, NV, 2013.

[Iyer *et al.*, 2013] R. Iyer, S. Jegelka, and J. Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. In *Advances in Neural Information Processing Systems 26 (NIPS'13)*, pages 2742–2750, Lake Tahoe, NV, 2013.

[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'03)*, 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 and Guestrin, 2005] A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions. *Technical Report No. CMU-CALD-05-103*, 2005.

[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.

[Lin and Bilmes, 2010] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'10)*, pages 912–920, Los Angeles, CA, 2010.

[Nemhauser and Wolsey, 1978] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[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'15)*, pages 1765–1773, Montreal, Canada, 2015.

[Qian *et al.*, 2016] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou. Parallel Pareto optimization for subset selection. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, pages 1939–1945, New York, NY, 2016.

[Rosenkrantz *et al.*, 1977] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[Vondrák, 2010] J. Vondrák. Submodularity and curvature: The optimal algorithm. *RIMS Kokyuroku Bessatsu B*, 23:253–266, 2010.

[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'16)*, pages 819–826, Phoenix, AZ, 2016.

[Zheng *et al.*, 2013] Y. Zheng, F. Liu, and H.-P. 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 (KDD'13)*, pages 1436–1444, Chicago, IL, 2013.