Distributed Pareto Optimization for Subset Selection

Chao Qian[1], **Guiying Li**[1], **Chao Feng**[1], **Ke Tang**[2]
[1] Anhui Province Key Lab of Big Data Analysis and Application,
University of Science and Technology of China, Hefei 230027, China
[2] Shenzhen Key Lab of Computational Intelligence, Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
chaoqian@ustc.edu.cn, {lgy147, chaofeng}@mail.ustc.edu.cn, tangk3@sustc.edu.cn

Abstract

The subset selection problem that selects a few items from a ground set arises in many applications such as maximum coverage, influence maximization, sparse regression, etc. The recently proposed POSS algorithm is a powerful approximation solver for this problem. However, POSS requires centralized access to the full ground set, and thus is impractical for large-scale real-world applications, where the ground set is too large to be stored on one single machine. In this paper, we propose a distributed version of POSS (DPOSS) with a bounded approximation guarantee. DPOSS can be easily implemented in the MapReduce framework. Our extensive experiments using Spark, on various real-world data sets with size ranging from thousands to millions, show that DPOSS can achieve competitive performance compared with the centralized POSS, and is almost always better than the state-of-the-art distributed greedy algorithm RANDGREEDI.

1 Introduction

The subset selection problem, which aims to select a subset of size at most k from a ground set of n items for maximizing some given objective function f, captures a wide variety of real-world applications, such as maximum coverage [Feige, 1998], influence maximization [Kempe *et al.*, 2003], sparse regression [Miller, 2002], active set selection [Rasmussen, 2004] and exemplar-based clustering [Dueck and Frey, 2007], to name a few. It is generally NP-hard, but the simple greedy algorithm, which iteratively selects one item with the largest marginal gain, was shown to be a good approximation solver. For examples, for a monotone submodular function f, the greedy algorithm achieves the optimal approximation guarantee of $(1 - 1/e)$ [Nemhauser *et al.*, 1978; Nemhauser and Wolsey, 1978]; for sparse regression where f can be non-submodular, it obtains the best-so-far approximation guarantee of $(1-e^{-\gamma})$ [Das and Kempe, 2011], where γ is the submodularity ratio.

Recently, a new approach Pareto Optimization for Subset Selection (POSS) has been shown superior to the greedy algorithm [Qian *et al.*, 2015; 2017b]. The idea of POSS is to reformulate the original subset selection problem as a bi-objective optimization problem that requires maximizing the given objective f and minimizing the subset size simultaneously. To solve this bi-objective problem, a randomized iterative procedure is employed, which randomly generates a new solution (i.e., subset) in each iteration. POSS was proved to achieve the same general approximation guarantee as the greedy algorithm, and was shown better on some subclasses [Das and Kempe, 2008]. Furthermore, it has achieved significantly better empirical performance on the applications of influence maximization and sparse regression.

To achieve a good performance, POSS requires running $2ek^2n$ ($e \approx 2.71828$ is Euler's number) iterations, which could be unsatisfactory for large k and n. Qian *et al.* [2016] thus further proposed a parallel version of POSS (called PPOSS), which generates multiple new solutions in parallel in each iteration instead of generating only one new solution. PPOSS can achieve linear speedup in the number of iterations while preserving the solution quality.

Note that the subset selection applications often come with massive data sets, e.g., the number of social network users in influence maximization and the number of variables in sparse regression can be millions. However, both POSS and PPOSS require centralized access to the full data set, which makes them impractical for large-scale real-world applications. The large-scale data set cannot be stored on one single machine, and must be distributed among a set of machines.

In this paper, we propose a distributed version of POSS (called DPOSS), which has a bounded approximation guarantee for subset selection with monotone objective functions. DPOSS uses a two-round divide and conquer strategy and can be easily implemented in the MapReduce framework. We conducted experiments using Spark on maximum coverage and sparse regression, two typical applications with the objective function being submodular and non-submodular, respectively. The results on real-world data sets with size ranging from thousands to millions show that DPOSS achieves performance close to the centralized POSS (the average approximation ratios on the two applications are at least 99.6% and

98.6%, respectively), and clearly outperforms the state-of-the-art distributed greedy algorithm RANDGREEDI [Mirzasoleiman *et al.*, 2013; Barbosa *et al.*, 2015].

We start the rest of the paper by introducing the subset selection problem and the POSS algorithm, respectively. In Section 4, we propose the DPOSS algorithm and give the theoretical analysis. Section 5 presents the empirical studies. The final section concludes this paper.

2 Subset Selection

Given a ground set $V = \{v_1, v_2, \ldots, v_n\}$, we study the functions $f : 2^V \to \mathbb{R}$ over subsets of V. A set function f is monotone if for any $S \subseteq T$, $f(S) \leq f(T)$. 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 $S \subseteq T \subseteq V$,

$$f(T) - f(S) \leq \textstyle\sum_{v \in T \setminus S} \big(f(S \cup v) - f(S)\big), \qquad (1)$$

or equivalently, for any $S \subseteq T \subseteq V$ and $v \notin T$,

$$f(S \cup v) - f(S) \geq f(T \cup v) - f(T). \qquad (2)$$

Note that we represent a singleton set $\{v\}$ by v for simplicity.

We then give two notions of "approximate submodularity", which measure to what extent a general set function f has the submodular property. The γ- and α-submodularity ratios are defined based on Eqs. (1) and (2), respectively. For a monotone set function f, it is easy to see that $0 \leq \gamma_{S,l}(f) \leq 1$, $0 \leq \alpha_f \leq 1$, and f is submodular iff $\gamma_{S,l}(f) = \alpha_f = 1$. For some concrete monotone non-submodular functions, lower bounds on γ and α were derived [Das and Kempe, 2011; Elenberg *et al.*, 2016; Bian *et al.*, 2017; Qian *et al.*, 2018]. When f is clear, we will use $\gamma_{S,l}$ and α for short.

Definition 1 (γ-Submodularity Ratio [Das and Kempe, 2011])**.** *The submodularity ratio of a set function $f : 2^V \to \mathbb{R}$ with respect to a set $S \subseteq V$ and a parameter $l \geq 1$ is*

$$\gamma_{S,l}(f) = \min_{L \subseteq S, T : |T| \leq l, T \cap L = \emptyset} \frac{\sum_{v \in T} (f(L \cup v) - f(L))}{f(L \cup T) - f(L)}.$$

Definition 2 (α-Submodularity Ratio [Zhang and Vorobeychik, 2016; Qian *et al.*, 2017a])**.** *The submodularity ratio of a set function $f : 2^V \to \mathbb{R}$ is*

$$\alpha_f = \min_{S \subseteq T \subseteq V, v \notin T} \frac{f(S \cup v) - f(S)}{f(T \cup v) - f(T)}.$$

The subset selection problem as presented in Definition 3 is to select a subset S of V such that a given objective f is maximized with the size constraint $|S| \leq k$, where $|\cdot|$ denotes the size of a set. For a monotone objective function f, the greedy algorithm, which iteratively adds one item with the largest marginal gain until k items are selected, can achieve an approximation guarantee of $(1 - e^{-\gamma_{S,k}})$ [Das and Kempe, 2011], where S is the subset output by the greedy algorithm. Particularly, when f is submodular (i.e., $\gamma_{S,k} = 1$), the approximation guarantee becomes $1 - e^{-1}$, which is optimal in general [Nemhauser and Wolsey, 1978].

Definition 3 (Subset Selection)**.** *Given all items $V = \{v_1, v_2, \ldots, v_n\}$, an objective function f and a budget k, it is to find a subset of at most k items maximizing f, i.e.,*

$$\arg\max_{S \subseteq V} f(S) \quad \textit{s.t.} \quad |S| \leq k. \qquad (3)$$

Here are two applications of subset selection, that will be investigated in this paper. Given a family of sets that cover a universe of elements, maximum coverage [Feige, 1998] as presented in Definition 4 is to select at most k sets whose union is maximal. It is easy to verify that f is monotone and submodular. Sparse regression [Miller, 2002] as presented in Definition 5 is to find a sparse approximation solution to the linear regression problem. Note that we do not distinguish S and its index set $\{i \mid v_i \in S\}$ for notational convenience, and we assume without loss of generality that all variables are normalized to have expectation 0 and variance 1. The objective function $R^2_{z,S}$ is monotone, but not necessarily submodular [Das and Kempe, 2011].

Definition 4 (Maximum Coverage)**.** *Given a set U of elements, a collection $V = \{v_1, v_2, \ldots, v_n\}$ of subsets of U, and a budget k, it is to find at most k sets from V maximizing the number of covered elements, i.e.,*

$$\arg\max_{S \subseteq V} f(S) = \Big|\bigcup_{v_i \in S} v_i\Big| \quad \textit{s.t.} \quad |S| \leq k.$$

Definition 5 (Sparse Regression)**.** *Given all observation variables $V = \{v_1, v_2, \ldots, v_n\}$, a predictor variable z and a budget k, it is to find at most k variables from V maximizing the* squared multiple correlation *[Diekhoff, 1992; Johnson and Wichern, 2007], i.e.,*

$$\arg\max_{S \subseteq V} R^2_{z,S} = 1 - \mathrm{MSE}_{z,S} \quad \textit{s.t.} \quad |S| \leq k,$$

where $\mathrm{MSE}_{z,S}$ denotes the mean squared error*, i.e.,*

$$\mathrm{MSE}_{z,S} = \min_{\boldsymbol{\alpha} \in \mathbb{R}^{|S|}} \mathbb{E}\left[\left(z - \textstyle\sum_{i \in S} \alpha_i v_i\right)^2\right].$$

3 The POSS Algorithm

Qian et al. [2015] proposed a new subset selection method by Pareto optimization, briefly called POSS. Let a binary vector $\boldsymbol{s} \in \{0,1\}^n$ represent a subset S of V, where $s_i = 1$ if S contains the item v_i and $s_i = 0$ otherwise. We will not distinguish $\boldsymbol{s} \in \{0,1\}^n$ and its corresponding subset for notational convenience. The POSS algorithm reformulates the original problem Eq. (3) as a bi-objective minimization problem

$$\arg\min_{\boldsymbol{s} \in \{0,1\}^n} \quad (f_1(\boldsymbol{s}), f_2(\boldsymbol{s})),$$

where

$$f_1(\boldsymbol{s}) = \begin{cases} +\infty, & |\boldsymbol{s}| \geq 2k \\ -f(\boldsymbol{s}), & \text{otherwise} \end{cases}, \quad f_2(\boldsymbol{s}) = |\boldsymbol{s}|.$$

That is, POSS maximizes the original objective function f and minimizes the subset size $|\boldsymbol{s}|$ simultaneously. Note that setting f_1 to $+\infty$ is to exclude overly infeasible solutions (i.e., subsets), the size of which is at least $2k$.

To compare two solutions in the bi-objective setting, both the two objective values have to be considered. For two solutions $\boldsymbol{s}$ and $\boldsymbol{s'}$, $\boldsymbol{s}$ *weakly dominates* $\boldsymbol{s'}$ (i.e., $\boldsymbol{s}$ is *better* than $\boldsymbol{s'}$, denoted as $\boldsymbol{s} \preceq \boldsymbol{s'}$) if $f_1(\boldsymbol{s}) \leq f_1(\boldsymbol{s'}) \wedge f_2(\boldsymbol{s}) \leq f_2(\boldsymbol{s'})$; $\boldsymbol{s}$ *dominates* $\boldsymbol{s'}$ (i.e., $\boldsymbol{s}$ is *strictly better*, denoted as $\boldsymbol{s} \prec \boldsymbol{s'}$) if

Algorithm 1 POSS Algorithm

Input: all items $V = \{v_1, v_2, \ldots, v_n\}$, an objective function $f : 2^V \to \mathbb{R}$ and a budget k
Parameter: the number T of iterations
Output: a subset of V with at most k items
Process:

1: Let $\boldsymbol{s} = \{0\}^n$ and $P = \{\boldsymbol{s}\}$.
2: Let $t = 0$.
3: **while** $t < T$ **do**
4: Select $\boldsymbol{s}$ from P uniformly at random.
5: Generate $\boldsymbol{s'}$ by flipping each bit of $\boldsymbol{s}$ with prob. $1/n$.
6: **if** $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec \boldsymbol{s'}$ **then**
7: $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{s'} \preceq \boldsymbol{z}\}) \cup \{\boldsymbol{s'}\}$.
8: **end if**
9: $t = t + 1$.
10: **end while**
11: **return** $\arg\max_{\boldsymbol{s} \in P, |\boldsymbol{s}| \le k} f(\boldsymbol{s})$

$\boldsymbol{s} \preceq \boldsymbol{s'}$ and either $f_1(\boldsymbol{s}) < f_1(\boldsymbol{s'})$ or $f_2(\boldsymbol{s}) < f_2(\boldsymbol{s'})$. But if neither $\boldsymbol{s} \preceq \boldsymbol{s'}$ nor $\boldsymbol{s'} \preceq \boldsymbol{s}$, they are *incomparable*.

As described in Algorithm 1, POSS uses a randomized iterative procedure to solve the bi-objective problem. It starts from the solution $\{0\}^n$ representing an empty set (line 1) and then iteratively tries to improve the solutions in the archive P (lines 3-10). In each iteration, a new solution $\boldsymbol{s'}$ is generated by randomly flipping bits of an archived solution $\boldsymbol{s}$ (line 5), which is uniformly randomly selected from the current P (line 4); if $\boldsymbol{s'}$ is not dominated by any archived solution in P (line 6), it will be added into P, and meanwhile those archived solutions weakly dominated by $\boldsymbol{s'}$ will be removed from P (line 7). After running T iterations, the best solution w.r.t. the original problem Eq. (3) is selected from P, i.e., the solution with the largest f value among those satisfying the size constraint in P is selected (line 11).

For subset selection with monotone objective functions, POSS using $\mathbb{E}[T] \le 2ek^2n$ was proved to achieve the same general approximation guarantee as the greedy algorithm [Qian *et al.*, 2015; 2017b]. Note that we use $\mathbb{E}[\cdot]$ to denote the expectation of a random variable. On some examples of maximum coverage and sparse regression, it was shown that POSS using polynomial iterations can find an optimal solution while the greedy algorithm cannot. Since the required number $2ek^2n$ of iterations is impractical for large k and n, Qian *et al.* [2016] further proposed the PPOSS algorithm by generating multiple solutions in parallel in each iteration instead of generating only one solution. They proved that PPOSS can achieve linear speedup in the number of iterations without sacrificing the solution quality. However, both POSS and PPOSS require centralized access to the ground set V, since randomly generated subsets of V have to be evaluated on one single machine. Thus, they are not readily applicable to large-scale subset selection applications, where the ground set V is too large to be stored on one single machine.

4 The DPOSS Algorithm

In this section, we propose a distributed version of POSS, called DPOSS. As shown in Algorithm 2, DPOSS is a simple two-round algorithm. In the first round, it arbitrarily distributes the ground set V over m machines, and then each machine runs POSS to find a subset $\boldsymbol{s_i}$ ($1 \le i \le m$) in parallel. In the second round, the m resulting subsets are merged on one machine, and then POSS is run on $\cup_{i=1}^m \boldsymbol{s_i}$ to find another subset $\boldsymbol{s_{m+1}}$. The final returned subset is the best one among these $m+1$ subsets. The number of iterations in each run of POSS is a parameter, which could affect the quality of the final output subset. Their relation will be analyzed later.

Algorithm 2 DPOSS Algorithm

Input: all items $V = \{v_1, v_2, \ldots, v_n\}$, an objective function $f : 2^V \to \mathbb{R}$, a budget k and the number m of machines
Parameter: $T_1, T_2, \ldots, T_m, T_{m+1}$
Output: a subset of V with at most k items
Process:

1: Partition V into m sets $V_1, V_2, \ldots, V_m$ arbitrarily so that each V_i can fit on one machine.
2: Run POSS with $T = T_i$ on each V_i to find a subset $\boldsymbol{s_i}$.
3: Merge the m resulting subsets into a set $U = \cup_{i=1}^m \boldsymbol{s_i}$.
4: Run POSS with $T = T_{m+1}$ on U to find a subset $\boldsymbol{s_{m+1}}$.
5: **return** $\arg\max_{\boldsymbol{s} \in \{\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots, \boldsymbol{s_{m+1}}\}} f(\boldsymbol{s})$

Note that when the size of the union $\cup_{i=1}^m \boldsymbol{s_i}$ exceeds the capacity of one machine in the second round of DPOSS, the algorithm will break down. But DPOSS can be easily extended to multi-round to address this issue. The idea is to continue to distribute the union of the partial subsets over several machines after each round, until the union can fit on one single machine. Note that if we have extra machines, we can also run PPOSS instead of POSS for further acceleration.

Then, we theoretically analyze the approximation performance of DPOSS for subset selection with monotone objective functions. Let OPT denote the optimal function value of Eq. (3). For an arbitrary partition $\{V_1, V_2, \ldots, V_m\}$ of the ground set V, let $n_i = |V_i|$ and $n_{\max} = \max\{n_i \mid 1 \le i \le m\}$. Theorem 1 gives the approximation guarantee of DPOSS. The proof is inspired from that of Theorem 1 in [Qian *et al.*, 2015], and it relies on the following two lemmas. For $1 \le i \le m$, let $\boldsymbol{o_i} \in \arg\max_{\boldsymbol{s} \subseteq V_i : |\boldsymbol{s}| \le k} f(\boldsymbol{s})$ denote an optimal subset of V_i. Lemma 1 gives a lower bound on the f value of the best $\boldsymbol{o_i}$. Its proof is inspired from that of Theorem 3 in [Mirzasoleiman *et al.*, 2016]. Lemma 2 shows that for any subset $\boldsymbol{s} \subseteq V_i$, there exists another item, the inclusion of which can improve f by at least a quantity proportional to the current distance to the optimum.

Lemma 1. *For any partition of V, it holds that*

$$\max\{f(\boldsymbol{o_i}) \mid 1 \le i \le m\} \ge \max\left\{\alpha/m, \gamma_{\emptyset,k}/k\right\} \cdot OPT.$$

Proof. We first prove that $\max\{f(\boldsymbol{o_i}) | 1 \le i \le m\} \ge \frac{\alpha}{m} OPT$. Let $\boldsymbol{o}$ denote an optimal subset of V, i.e., $f(\boldsymbol{o}) = OPT$. For $1 \le i \le m$, let $A_i = \boldsymbol{o} \cap V_i$. Thus, $\cup_{i=1}^m A_i = \boldsymbol{o}$ and for any $i \ne j$, $A_i \cap A_j = \emptyset$. Then, we have

$$f(\boldsymbol{o}) = f(\cup_{i=1}^m A_i) = \sum_{i=1}^m f(\cup_{j=1}^i A_j) - f(\cup_{j=1}^{i-1} A_j).$$

Let $\{v_1^i, v_2^i, \ldots, v_{|A_i|}^i\}$ denote the items in A_i. Then, for any

$1 \le i \le m$, it holds that

$$f(\cup_{j=1}^{i} A_j) - f(\cup_{j=1}^{i-1} A_j)$$
$$= \sum_{l=1}^{|A_i|} f(\cup_{j=1}^{i-1} A_j \cup \{v_1^i, \ldots, v_l^i\}) - f(\cup_{j=1}^{i-1} A_j \cup \{v_1^i, \ldots, v_{l-1}^i\})$$
$$\le \frac{1}{\alpha} \sum_{l=1}^{|A_i|} f(\{v_1^i, \ldots, v_l^i\}) - f(\{v_1^i, \ldots, v_{l-1}^i\}) = \frac{f(A_i)}{\alpha},$$

where the inequality is by the definition of α-submodularity ratio (i.e., Definition 2) since $\{v_1^i, \ldots, v_{l-1}^i\} \subseteq \cup_{j=1}^{i-1} A_j \cup \{v_1^i, \ldots, v_{l-1}^i\}$. Note that for any $1 \le i \le m$, $f(\boldsymbol{o_i}) \ge f(A_i)$, since $A_i \subseteq V_i$ and $|A_i| \le |\boldsymbol{o}| \le k$. Thus, we get

$$OPT = f(\boldsymbol{o}) \le \frac{1}{\alpha} \sum_{i=1}^{m} f(A_i) \le \frac{1}{\alpha} \sum_{i=1}^{m} f(\boldsymbol{o_i}),$$

which leads to $\max\{f(\boldsymbol{o_i}) \mid 1 \le i \le m\} \ge \frac{\alpha}{m} \cdot OPT$.

We then prove that $\max\{f(\boldsymbol{o_i}) | 1 \le i \le m\} \ge \frac{\gamma_{\emptyset,k}}{k} OPT$. By the definition of γ-submodularity ratio (i.e., Definition 1), $f(\boldsymbol{o}) \le \sum_{v \in \boldsymbol{o}} f(v)/\gamma_{\emptyset,k}$. Let $v^* \in \arg\max_{v \in \boldsymbol{o}} f(v)$. Then $f(v^*) \ge \frac{\gamma_{\emptyset,k} f(\boldsymbol{o})}{|\boldsymbol{o}|} \ge \frac{\gamma_{\emptyset,k}}{k} \cdot OPT$. Since $\{V_1, V_2, \ldots, V_m\}$ is a partition of V, v^* must belong to one of these m sets. Thus, $\max\{f(\boldsymbol{o_i}) \mid 1 \le i \le m\} \ge f(v^*) \ge \frac{\gamma_{\emptyset,k}}{k} \cdot OPT$. □

Lemma 2. *[Qian* et al.*, 2016] For any $\boldsymbol{s} \subseteq V_i$ $(1 \le i \le m)$, there exists one item $v \in V_i \setminus \boldsymbol{s}$ such that*

$$f(\boldsymbol{s} \cup v) - f(\boldsymbol{s}) \ge (\gamma_{\boldsymbol{s},k}/k) \cdot (f(\boldsymbol{o_i}) - f(\boldsymbol{s})).$$

Theorem 1. *For subset selection with a monotone objective function f, DPOSS using $\mathbb{E}[\max\{T_i \mid 1 \le i \le m\}] = O(k^2 n_{\max}(1 + \log m))$ finds a subset $\boldsymbol{s}$ with $|\boldsymbol{s}| \le k$ and*

$$f(\boldsymbol{s}) \ge (1 - e^{-\gamma_{\min}}) \cdot \max\left\{\alpha/m, \gamma_{\emptyset,k}/k\right\} \cdot OPT,$$

where $\gamma_{\min} = \min_{\boldsymbol{s} \subseteq V : |\boldsymbol{s}| = k-1} \gamma_{\boldsymbol{s},k}$.

Proof. In the first round of DPOSS, we analyze the maximum number of iterations (i.e., $\max\{T_i \mid 1 \le i \le m\}$) on each machine until $f(\boldsymbol{s_i}) \ge (1 - e^{-\gamma_{\min}}) \cdot f(\boldsymbol{o_i})$ for each $1 \le i \le m$. For the machine running POSS on V_i $(1 \le i \le m)$, let $J_{\max}^i$ denote the maximum value of $j \in \{0, 1, \ldots, k\}$ such that in the archive P, there exists a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \le j$ and $f(\boldsymbol{s}) \ge (1 - (1 - \frac{\gamma_{\min}}{k})^j) \cdot f(\boldsymbol{o_i})$. That is,

$$J_{\max}^i = \max\{j \in \{0, 1, \ldots, k\} \mid \exists \boldsymbol{s} \in P, |\boldsymbol{s}| \le j \ \wedge \ f(\boldsymbol{s}) \ge (1 - (1 - \gamma_{\min}/k)^j) \cdot f(\boldsymbol{o_i})\}.$$

We then only need to analyze $\max\{T_i \mid 1 \le i \le m\}$ until $\min\{J_{\max}^i \mid 1 \le i \le m\} = k$, since $J_{\max}^i = k$ implies that for POSS running on V_i, there exists one solution $\boldsymbol{s}$ in P satisfying that $|\boldsymbol{s}| \le k$ and $f(\boldsymbol{s}) \ge (1 - (1 - \frac{\gamma_{\min}}{k})^k) \cdot f(\boldsymbol{o_i}) \ge (1 - e^{-\gamma_{\min}}) \cdot f(\boldsymbol{o_i})$, thus $f(\boldsymbol{s_i}) \ge (1 - e^{-\gamma_{\min}}) \cdot f(\boldsymbol{o_i})$.

The initial value of $\min\{J_{\max}^i \mid 1 \le i \le m\}$ is 0, since for any $1 \le i \le m$, POSS running on V_i starts from $\{0\}^n$, and then $J_{\max}^i = 0$. Assume that currently $\min\{J_{\max}^i \mid 1 \le i \le m\} = j < k$ and the number of $J_{\max}^i = j$ is l $(1 \le l \le m)$, i.e., $|\{J_{\max}^i = j \mid 1 \le i \le m\}| = l$. For POSS running on V_i, let $\boldsymbol{z_i}$ be a corresponding solution with the value $J_{\max}^i$, i.e., $|\boldsymbol{z_i}| \le J_{\max}^i$ and

$$f(\boldsymbol{z_i}) \ge \left(1 - (1 - \gamma_{\min}/k)^{J_{\max}^i}\right) \cdot f(\boldsymbol{o_i}). \tag{4}$$

It is easy to see that $J_{\max}^i$ cannot decrease because cleaning $\boldsymbol{z_i}$ from P (line 7 of Algorithm 1) implies that $\boldsymbol{z_i}$ is weakly dominated by the newly generated solution, which must have a smaller size and a larger f value. Thus, we can conclude that j never decreases and the corresponding l never increases.

By Lemma 2, we know that for any $1 \le i \le m$, flipping one specific 0 bit of $\boldsymbol{z_i}$ (i.e., adding a specific item) can generate a new solution $\boldsymbol{s'}$, which satisfies that $f(\boldsymbol{s'}) - f(\boldsymbol{z_i}) \ge \frac{\gamma_{\boldsymbol{z_i},k}}{k}(f(\boldsymbol{o_i}) - f(\boldsymbol{z_i}))$. Then, if $J_{\max}^i < k$, we have

$$f(\boldsymbol{s'}) \ge (1 - \gamma_{\boldsymbol{z_i},k}/k) f(\boldsymbol{z_i}) + (\gamma_{\boldsymbol{z_i},k}/k) \cdot f(\boldsymbol{o_i}) \ge \left(1 - (1 - \gamma_{\min}/k)^{J_{\max}^i + 1}\right) \cdot f(\boldsymbol{o_i}),$$

where the last inequality is by Eq. (4) and $\gamma_{\boldsymbol{z_i},k} \ge \gamma_{\min}$, which can be easily derived from $|\boldsymbol{z_i}| \le J_{\max}^i < k$ and $\gamma_{\boldsymbol{s},k}$ decreasing with $\boldsymbol{s}$. Since $|\boldsymbol{s'}| = |\boldsymbol{z_i}| + 1 \le J_{\max}^i + 1$, $\boldsymbol{s'}$ will be included into P; otherwise, $\boldsymbol{s'}$ must be dominated by one solution in P (line 6 of Algorithm 1), which makes a contradiction with the definition of $J_{\max}^i$. After including $\boldsymbol{s'}$, $J_{\max}^i$ increases by at least 1. Let $P_{\max}$ denote the largest size of P during the run of POSS. Thus, $J_{\max}^i$ can increase by at least 1 in one iteration with probability at least $\frac{1}{P_{\max}} \cdot \frac{1}{n_i}(1 - \frac{1}{n_i})^{n_i - 1} \ge \frac{1}{e n_i P_{\max}}$, where $\frac{1}{P_{\max}}$ is a lower bound on the probability of selecting $\boldsymbol{z_i}$ in line 4 of Algorithm 1 and $\frac{1}{n_i}(1 - \frac{1}{n_i})^{n_i - 1}$ is the probability of flipping only a specific bit of $\boldsymbol{z_i}$ in line 5. By the procedure of POSS, we know that the solutions maintained in P must be incomparable. Thus, each value of one objective can correspond to at most one solution in P. Because the solutions with $|\boldsymbol{s}| \ge 2k$ have $+\infty$ value on the first objective, they must be excluded from P. Thus, $P_{\max} \le 2k$, which implies that $J_{\max}^i$ can increase by at least 1 in one iteration with probability at least $\frac{1}{2ekn_i}$. We then get that after one iteration in the first round of DPOSS, l can decrease by at least 1 with probability at least

$$1 - \prod_{i : J_{\max}^i = j} \left(1 - \frac{1}{2ekn_i}\right) \ge 1 - \left(1 - \frac{1}{2ekn_{\max}}\right)^l,$$

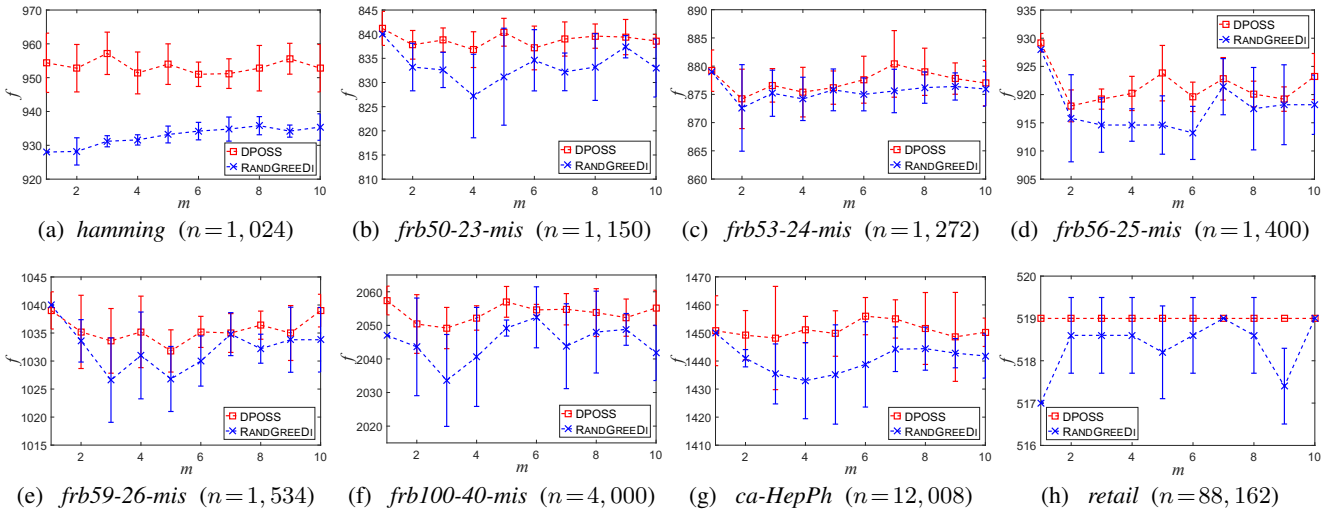
since it is sufficient that at least one of those $J_{\max}^i = j$ increases. Thus, the expected number of iterations until j increases (i.e., l decreases to 0) is at most

$$\sum_{l=1}^{m} \frac{1}{1 - \left(1 - \frac{1}{2ekn_{\max}}\right)^l} = \sum_{l=1}^{m} 1 + \frac{1}{\left(1 - \frac{1}{2ekn_{\max}}\right)^l - 1} \le \sum_{l=1}^{m} 1 + \frac{2ekn_{\max} - 1}{l} = m + (2ekn_{\max} - 1)H_m,$$

where the inequality is by $\frac{1}{(1 - \frac{1}{2ekn_{\max}})^l} = (1 + \frac{\frac{1}{2ekn_{\max}}}{1 - \frac{1}{2ekn_{\max}}})^l \ge 1 + \frac{\frac{l}{2ekn_{\max}}}{1 - \frac{1}{2ekn_{\max}}} = 1 + \frac{l}{2ekn_{\max} - 1}$, and H_m is the m-th harmonic number. Then, the expected number of iterations until $\min\{J_{\max}^i \mid 1 \le i \le m\} = k$ (i.e., j increases to k) is at most $(m + (2ekn_{\max} - 1)H_m)k = O(k^2 n_{\max}(1 + \log m))$, i.e.,

$$\mathbb{E}[\max\{T_i \mid 1 \le i \le m\}] = O(k^2 n_{\max}(1 + \log m)).$$

Since $\min\{J_{\max}^i \mid 1 \le i \le m\} = k$ implies that $f(\boldsymbol{s_i}) \ge (1 - e^{-\gamma_{\min}}) \cdot f(\boldsymbol{o_i})$ for any $1 \le i \le m$, the f value of the final output subset satisfies that $\max\{f(\boldsymbol{s_i}) \mid 1 \le i \le m + 1\} \ge (1 - e^{-\gamma_{\min}}) \cdot \max\{f(\boldsymbol{o_i}) \mid 1 \le i \le m\}$. By Lemma 1, the theorem holds. □



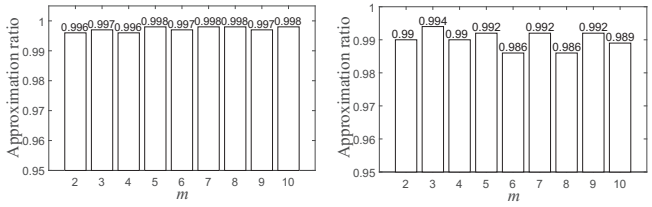
(a) *hamming* ($n=1,024$) (b) *frb50-23-mis* ($n=1,150$) (c) *frb53-24-mis* ($n=1,272$) (d) *frb56-25-mis* ($n=1,400$)

(e) *frb59-26-mis* ($n=1,534$) (f) *frb100-40-mis* ($n=4,000$) (g) *ca-HepPh* ($n=12,008$) (h) *retail* ($n=88,162$)

Figure 1: The comparison between DPOSS and RANDGREEDI on 8 regular-scale data sets of maximum coverage (f: the number of covered elements, the larger the better). For each data set, n denotes the total number of subsets.

Data set	DPOSS	RANDGREEDI
accident ($n=340,183$)	175±1	170.6±1.34
kosarak ($n=990,002$)	9263±0	9263±0

Table 1: The f value (mean±std.) of DPOSS and RANDGREEDI on large-scale data sets of maximum coverage (f: the number of covered elements, the larger the better).

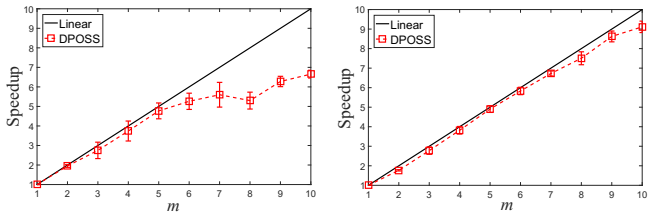


(a) maximum coverage (b) sparse regression

Figure 2: The approximation ratio of DPOSS compared to the centralized POSS, averaged over all regular-scale data sets.

5 Experiments

In this section, we empirically evaluate the effectiveness of DPOSS on maximum coverage and sparse regression, two applications of subset selection with submodular and non-submodular objective functions, respectively. All of the experiments are coded in Python 2.7.14 and run on an identical configuration: a cluster of 10 quad-core machines running Spark 2.0.2. We compare DPOSS with the state-of-the-art distributed greedy algorithm RANDGREEDI [Mirzasoleiman *et al.*, 2013; Barbosa *et al.*, 2015; Mirrokni and Zadimoghaddam, 2015; Lucic *et al.*, 2016; Khanna *et al.*, 2017] on regular-scale as well as large-scale data sets. By regular-scale data sets, we can examine how well DPOSS performs compared with the centralized POSS. To implement DPOSS, each machine runs POSS for $2ek^2N$ iterations, where N is the number of items allocated to that machine, as suggested in [Qian *et al.*, 2015; 2017b].

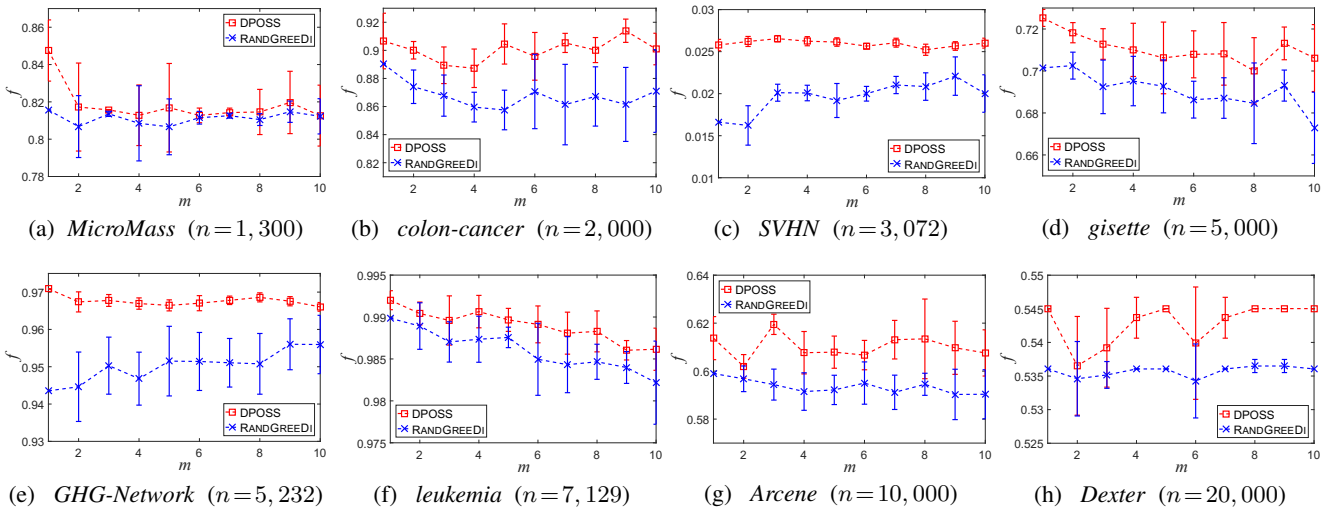


(a) *hamming* (b) *frb100-40-mis*

Figure 3: The runtime speedup of DPOSS compared to the centralized POSS on maximum coverage.

Maximum Coverage. We use 8 regular-scale and 2 large-scale data sets[1]. Note that some data sets are given by graphs, and we create a set for each node which contains the node itself and its adjacent nodes. The budget k is set to 8. For regular-scale data sets, we set the number m of reducers to $\{1, 2, \ldots, 10\}$. For each data set and each $m > 1$, we perform the partition by assigning items uniformly randomly to the reducers. We run DPOSS and RANDGREEDI on the same 10 partitions generated independently and report the average results. When $m = 1$, we also repeat the run of the centralized POSS 10 times independently, since it is a randomized algorithm. Note that DPOSS with $m = 1$ is just the centralized POSS. The results are plotted in Figure 1. We can observe that DPOSS is almost always better than RANDGREEDI, and the only loss is for $m = 1$ on the *frb59-26-mis* data set. Note that on the *retail* data set, the standard deviation of the f value by DPOSS is 0, which is because DPOSS always finds the same good solution in 10 runs for each m. For large-scale data sets, m is set to 300, and each machine carries out a set

[1]The data sets are downloaded from `http://fimi.ua.ac.be/data/`, `https://snap.stanford.edu/data/`, `https://turing.cs.hbg.psu.edu/txn131/vertex_cover.html` and `http://sites.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm`.



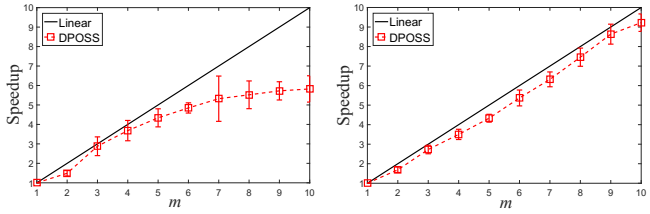
(a) *MicroMass* ($n=1,300$) (b) *colon-cancer* ($n=2,000$) (c) *SVHN* ($n=3,072$) (d) *gisette* ($n=5,000$)

(e) *GHG-Network* ($n=5,232$) (f) *leukemia* ($n=7,129$) (g) *Arcene* ($n=10,000$) (h) *Dexter* ($n=20,000$)

Figure 4: The comparison between DPOSS and RANDGREEDI on 8 regular-scale data sets of sparse regression (f: the squared multiple correlation R^2, the larger the better). For each data set, n denotes the total number of observation variables.

Data set	DPOSS	RANDGREEDI
Gas-sensor-flow ($n=120,432$)	.818±.005	.710±.017
Twin-gas-sensor ($n=480,000$)	.601±.014	.470±.025
Gas-sensor-sample ($n=1,950,000$)	.289±.029	.245±.018

Table 2: The f value (mean±std.) of DPOSS and RANDGREEDI on large-scale data sets of sparse regression (f: the squared multiple correlation R^2, the larger the better).



(a) *MicroMass* (b) *leukemia*

Figure 5: The runtime speedup of DPOSS compared to the centralized POSS on sparse regression.

of reduce tasks in sequence. The results are shown in Table 1. We can observe that DPOSS performs the same as RANDGREEDI on the *kosarak* data set and is better on the other data set *accident*. The same performance on *kosarak* may be because RANDGREEDI has already been nearly optimal on this data set, as observed in [Barbosa *et al.*, 2015].

For regular-scale data sets, we also compute the approximation ratio of DPOSS compared to the centralized POSS. For each $m = i \in \{2,\ldots,10\}$, the approximation ratio is calculated through dividing the f value for $m = i$ by that for $m = 1$. Figure 2(a) plots the average approximation ratios over all data sets, which are at least 99.6%, implying that DPOSS can achieve performance very close to the centralized POSS. From Figure 1, we can also observe that in some cases (e.g., $m = 3$ on the *hamming* data set), DPOSS can even be better than the centralized POSS, which may be because the partition of the full data set luckily avoids the local optimum.

Figure 3 shows the runtime speedup of DPOSS compared to the centralized POSS on two data sets *hamming* and *frb100-40-mis*. For each $m=i\in\{2,\ldots,10\}$, the speedup is calculated as the ratio of the runtime for $m=1$ and $m=i$. For $m=1$, the centralized POSS runs for $2ek^2n$ iterations. For $m>1$, the number of iterations of DPOSS consists of two parts: nearly $2ek^2n/m$ on each machine in the first round due to uniform random partition, and $2ek^2\cdot(mk)$ in the second round. Thus, when m is much smaller than $\sqrt{n/k}$, the runtime of the second round can be neglected and DPOSS can achieve nearly linear speedup; when m continues to increase, the runtime of the second round will gradually dominate the whole runtime and the speedup will be far away from linear speedup. The results in Figure 3 are consistent with the analysis. For *hamming*, $\sqrt{n/k}=11.3$ and thus we cannot observe linear speedup as m approaches to 10; while for *frb100-40-mis*, $\sqrt{n/k}=22.4$ is much larger than the maximum $m=10$ and thus DPOSS always achieves nearly linear speedup.

Sparse Regression. We use 8 regular-scale and 3 large-scale data sets[2]. Note that some classification data sets are used for regression, and all variables are normalized to have mean 0 and variance 1. We use the same setting as that for maximum coverage. The results on regular-scale and large-scale data sets are shown in Figure 4 and Table 2, respectively. We can see that DPOSS is always better than RANDGREEDI. Figure 2(b) plots the average approximation ratios of DPOSS compared to the centralized POSS over all regular-scale data sets, which are at least 98.6%. We also plot the runtime speedup of DPOSS on two data sets *MicroMass* and *leukemia* in Figure 5, which is similar to that we have observed for maximum coverage.

[2]The data sets are downloaded from `https://archive.ics.uci.edu/ml/datasets.html` and `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

6 Conclusion

Subset selection is a fundamental problem in many areas, and the POSS algorithm has been shown to be a powerful approximation solver. However, POSS requires centralized access to the full data set, restricting its large-scale applications. In this paper, we propose a distributed version of POSS (DPOSS) with a bounded approximation guarantee. Extensive experiments using Spark on the applications of maximum coverage and sparse regression show that DPOSS can achieve competitive performance to the centralized POSS; can scale well to very large data sets; and can clearly outperform the state-of-the-art distributed greedy algorithm RANDGREEDI.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (2017YFB1003102), the NSFC (61603367, 61672478), the Science and Technology Innovation Committee Foundation of Shenzhen (ZDSYS201703031748284), and the Royal Society Newton Advanced Fellowship (NA150123).

References

[Barbosa *et al.*, 2015] R. Barbosa, A. Ene, H. Nguyen, and J. Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *ICML*, pages 1236–1244, Lille, France, 2015.

[Bian *et al.*, 2017] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. In *ICML*, pages 498–507, Sydney, Australia, 2017.

[Das and Kempe, 2008] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *STOC*, pages 45–54, Victoria, Canada, 2008.

[Das and Kempe, 2011] A. Das and D. Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, pages 1057–1064, Bellevue, WA, 2011.

[Diekhoff, 1992] G. Diekhoff. *Statistics for the Social and Behavioral Sciences: Univariate, Bivariate, Multivariate*. William C Brown Pub, 1992.

[Dueck and Frey, 2007] D. Dueck and B. J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *ICCV*, pages 1–8, Rio de Janeiro, Brazil, 2007.

[Elenberg *et al.*, 2016] E. R. Elenberg, R. Khanna, A. G. Dimakis, and S. Negahban. Restricted strong convexity implies weak submodularity. *arXiv:1612.00804*, 2016.

[Feige, 1998] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[Johnson and Wichern, 2007] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Pearson, 6th edition, 2007.

[Kempe *et al.*, 2003] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, Washington, DC, 2003.

[Khanna *et al.*, 2017] R. Khanna, E. Elenberg, A. Dimakis, S. Negahban, and J. Ghosh. Scalable greedy feature selection via weak submodularity. In *AISTATS*, pages 1560–1568, Fort Lauderdale, FL, 2017.

[Lucic *et al.*, 2016] M. Lucic, O. Bachem, M. Zadimoghaddam, and A. Krause. Horizontally scalable submodular maximization. In *ICML*, pages 2981–2989, New York City, NY, 2016.

[Miller, 2002] A. Miller. *Subset Selection in Regression*. Chapman and Hall/CRC, 2nd edition, 2002.

[Mirrokni and Zadimoghaddam, 2015] V. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *STOC*, pages 153–162, Portland, OR, 2015.

[Mirzasoleiman *et al.*, 2013] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*, pages 2049–2057, Lake Tahoe, NV, 2013.

[Mirzasoleiman *et al.*, 2016] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization. *Journal of Machine Learning Research*, 17(238):1–44, 2016.

[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 *NIPS*, 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 *IJCAI*, pages 1939–1945, New York, NY, 2016.

[Qian *et al.*, 2017a] C. Qian, J.-C. Shi, Y. Yu, and K. Tang. On subset selection with general cost constraints. In *IJCAI*, pages 2613–2619, Melbourne, Australia, 2017.

[Qian *et al.*, 2017b] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou. Subset selection under noise. In *NIPS*, pages 3562–3572, Long Beach, CA, 2017.

[Qian *et al.*, 2018] C. Qian, Y. Yu, and K. Tang. Approximation guarantees of stochastic greedy algorithms for subset selection. In *IJCAI*, Stockholm, Sweden, 2018.

[Rasmussen, 2004] C. E. Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[Zhang and Vorobeychik, 2016] H. Zhang and Y. Vorobeychik. Submodular optimization with routing constraints. In *AAAI*, pages 819–826, Phoenix, AZ, 2016.