Distributed Pareto Optimization for Large-scale Noisy Subset Selection

Chao Qian, *Member, IEEE*

***Abstract*—Subset selection, aiming to select the best subset from a ground set with respect to some objective function, is a fundamental problem with applications in many areas, such as combinatorial optimization, machine learning, data mining, computer vision, information retrieval, etc. Along with the development of data collection and storage, the size of the ground set grows larger. Furthermore, in many subset selection applications, the objective function evaluation is subject to noise. We thus study the large-scale noisy subset selection problem in this paper. The recently proposed DPOSS algorithm based on multi-objective evolutionary optimization is a powerful distributed solver for large-scale subset selection. Its performance, however, has been only validated in the noise-free environment. In this paper, we first prove its approximation guarantee under two common noise models, i.e., multiplicative noise and additive noise, disclosing that the presence of noise degrades the performance of DPOSS largely. Next, we propose a new distributed multi-objective evolutionary algorithm called DPONSS for large-scale noisy subset selection. We prove that the approximation guarantee of DPONSS under noise is significantly better than that of DPOSS. We also conduct experiments on the application of sparse regression, where the objective evaluation is often estimated using a sample data, bringing noise. The results on various real-world data sets, whose size can reach millions, clearly show the excellent performance of DPONSS.**

***Index Terms*—Subset selection, large-scale, noise, Pareto optimization, multi-objective evolutionary algorithms, distributed algorithms, theoretical analyses, experimental studies.**

I. Introduction

In real-world applications, we often encounter the problem of selecting a good subset from a total set of items. For example, for influence maximization [22], it is to select a subset of social network users which has a large influence spread; for sensor placement [23], it is to select a subset of places to install sensors such that the information gathered is maximized or the uncertainty of the environment is mostly reduced; for sparse regression [26], it is to select a subset of observation variables to best approximate the target variable by linear regression; for unsupervised feature selection [13], it is to select a subset of features to minimize the reconstruction error of the data matrix. All of these applications can be formulated as such a constrained optimization problem:

$$\arg\max_{S\subseteq V} f(S) \quad s.t. \quad |S| \leq k, \tag{1}$$

where $V = \{v_1, v_2, \ldots, v_n\}$ is a ground set of n items, f is a given objective function to be maximized, $|\cdot|$ denotes the size of a set, and k is the budget. Each application has specific meanings on the items $\{v_1, v_2, \ldots, v_n\}$ and the objective f. This general problem is called *subset selection*, the goal of which is to select a subset with at most k items for maximizing a given objective function f. More applications include maximum coverage [14], active set selection [39], exemplar-based clustering [11], etc.

Therefore, subset selection is a very general problem with many practical applications. Due to its NP-hardness [28], several polynomial-time approximation algorithms have been designed. The most famous one is the standard greedy algorithm. Starting from the empty set, the greedy algorithm iteratively adds one item which brings the largest increment on the objective f, until k items have been selected. It has been proved that when the objective function f is monotone, the greedy algorithm can achieve an approximation ratio of $(1 - e^{-\gamma})$ [7], where γ is the submodularity ratio measuring how close f is to submodular. Particularly, for the cases where f is submodular, $\gamma = 1$ and thus the approximation ratio becomes $(1 - 1/e)$, which is optimal [29], [30]; for sparse regression where f is not necessarily submodular, the approximation ratio $(1 - e^{-\gamma})$ is also the best known one [7].

A simple local search (LS) algorithm [30] is also often used for subset selection. Starting from k randomly selected items, LS iteratively replaces a selected item with an unselected one, which can bring improvement on the objective f. This process is terminated until no improvement can be yielded. It has been shown [30] that for monotone submodular f, LS can achieve an approximation ratio of $k/(2k-1)$, which is, however, worse than that, i.e., $1 - 1/e$, of the greedy algorithm.

Evolutionary algorithms (EAs), as a kind of global search algorithms, have achieved great successes in solving complicated optimization problems, e.g., [4], [10], [31], [32], [40], [42]. Based on multi-objective evolutionary optimization, Qian et al. [36] recently proposed a new method Pareto Optimization for Subset Selection (POSS). POSS first reformulates the original problem, i.e., Eq. (1), as a bi-objective optimization problem: maximizing the given objective $f(S)$ and minimizing the subset size $|S|$. Then, a simple multi-objective EA (MOEA) with mutation only is employed to solve this bi-objective optimization problem. Finally, the best feasible solution, i.e., the solution having the largest f value while satisfying the size constraint, is selected from the generated non-dominated solution set. Note that a solution here corresponds to a subset. POSS has been proved to be able to obtain the same general approximation guarantee $(1 - e^{-\gamma})$ as the greedy algorithm. Furthermore, it has been empirically shown significantly better than the greedy algorithm and LS on the application of sparse regression [36] and unsupervised feature selection [15].

C. Qian is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China, and also with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: qianc@lamda.nju.edu.cn)

As the development of data collection and storage techniques, the size of the ground set V involved in the subset selection applications becomes larger and larger. For example, for the application of influence maximization, a social network can contain millions of users; for sparse regression or unsupervised feature selection, the number of variables can also reach millions, or even tens of millions. However, the application of POSS to large-scale subset selection tasks is limited. In each iteration of POSS, new offspring solutions are randomly generated by mutation and then evaluated. Thus, running POSS requires centralized access to the whole V, whereas the data set can be too large to be stored on one single machine for large-scale applications.

To alleviate the limitation of POSS to large-scale applications, a distributed version called DPOSS has been proposed [33]. DPOSS is a two-round algorithm and employs a divide and conquer strategy. That is, the ground set V is partitioned and distributed over m machines, and then POSS is run on each machine to find a subset in parallel; after this, another subset is generated by running POSS on the union of these m resulting subsets. DPOSS finally outputs the best one among all the $(m+1)$ subsets generated in these two rounds. On the two large-scale applications, i.e., maximum coverage and sparse regression, it has been shown that the performance of DPOSS is very close to that of the centralized POSS algorithm, and clearly better than that of the state-of-the-art distributed greedy algorithm RANDGREEDI [1], [27]. Furthermore, it has been proved that for the subset selection problem having monotone objective functions, DPOSS has an approximation guarantee of $(1-e^{-\gamma})\cdot\max\{\alpha/m,\gamma/k\}$. Note that α is another notion of approximate submodularity [43], different from γ.

Thus, DPOSS is a powerful approximation solver for large-scale subset selection. However, its performance has only been validated in the noise-free environment, whereas the objective function evaluation is often subject to noise for large-scale subset selection. That is, only a noisy objective function value can be obtained instead of the exact one. The noise generally stems from two aspects. On one hand, the objective function evaluation can be very time-consuming and thus is often approximately calculated in some applications. For example, in the task of influence maximization, the computation of the objective, i.e., influence spread, is #P-hard [5], which is often estimated by simulating the random propagation process multiple times and using the average [22]; in the task of sparse regression, a sample of instances rather than all instances is often used for regression, leading to a noisy evaluation. On the other hand, the noise can be produced by the distributed algorithm. For some applications, e.g., exemplar-based clustering [11] and unsupervised feature selection [13], the objective function relies on the whole data set V, whereas by the distribution, it can only be estimated based on the local data allocated to each machine.

In this paper, we first theoretically study the performance of DPOSS for large-scale subset selection under noise. Two common noise models, i.e., multiplicative noise and additive noise, are considered, where the noisy objective $F(S)$ is in the range of $(1\pm\epsilon)\cdot f(S)$ and $f(S)\pm\epsilon$, respectively. We prove that for subset selection having monotone f under multiplicative noise, DPOSS can achieve an approximation guarantee of

$$\frac{1}{1+\frac{2k\epsilon}{(1-\epsilon)\gamma}}\cdot\left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^{k}e^{-\gamma}\right)\cdot\max\left\{\frac{\alpha}{m},\frac{\gamma}{k}\right\}. \quad (2)$$

Compared with that without noise, i.e., $(1-e^{-\gamma})\cdot\max\{\alpha/m,\gamma/k\}$ [33], we find that the performance of DPOSS degrades largely due to the presence of noise. For example, for the case of submodular f where $\alpha=\gamma=1$, DPOSS without noise can achieve an approximation guarantee of $\Theta(\max\{1/m,1/k\})$, whereas such a guarantee can be obtained only when $\epsilon=O(1/k)$ under multiplicative noise. Under additive noise, we also prove that the approximation bound of DPOSS is much worse than that without noise. Note that there are several pieces of works [18], [19], [41] addressing noisy subset selection, but the algorithms cannot be directly applied to large-scale applications.

Next, we propose a new distributed algorithm called DPONSS for large-scale subset selection under noise. Similarly to DPOSS, DPONSS also uses the strategy of divide and conquer. The difference is that DPONSS runs the PONSS algorithm [34] on each machine, whereas DPOSS runs POSS. PONSS is modified from POSS by introducing a noise-aware comparison mechanism, which can bring the robustness against noise. Under multiplicative noise, we prove that DPONSS can achieve an approximation guarantee of

$$\frac{1-\epsilon}{1+\epsilon}\cdot(1-e^{-\gamma})\cdot\max\left\{\frac{\alpha}{m},\frac{\gamma}{k}\right\}, \quad (3)$$

based on some assumption like independently and identically distributed (i.i.d.) noise distribution. By comparing Eq. (3) with Eq. (2), we find that DPONSS achieves a much better approximation bound than DPOSS. For example, for the case of submodular f and constant ϵ, DPONSS obtains an approximation ratio of $\Theta(\max\{1/m,1/k\})$, whereas DPOSS can only guarantee a $\Theta((1/k)\cdot\max\{1/m,1/k\})$-approximation ratio. Under additive noise, we also prove that DPONSS obtains a better approximation bound than DPOSS.

Finally, we empirically compare DPONSS with the two state-of-the-art distributed algorithms, DPOSS and RANDGREEDI, on the application of sparse regression. We also implement the distributed version of LS, called DLS. All of them are easily implemented in the Hadoop MapReduce framework, and Spark is used in our experiments. For sparse regression, to estimate the regression error of a subset of variables, only a random sample of instances is used, which thus brings noise. Besides 9 large-scale data sets where the size can be millions, we also use 12 regular-scale data sets, by which we can examine whether the performance of DPONSS will decrease, compared to the centralized PONSS algorithm. The results on these real-world data sets clearly show that DPONSS performs the best, and DPOSS is the runner-up. Furthermore, the average approximation ratio of DPONSS over the centralized PONSS algorithm on regular-scale data sets is at least 97.5%, disclosing that the distribution implementation leads to little performance loss.

The rest of the paper is organized as follows. Section II first introduces some preliminaries on the studied problem,

i.e., large-scale noisy subset selection. In Section III, the performance of DPOSS under noise is analyzed theoretically. Section IV then presents the proposed DPONSS algorithm and its theoretical analysis. Section V gives the empirical study. Section VI finally concludes this paper.

II. Preliminaries

This section first introduces the subset selection problem and the POSS algorithm; then presents the distributed version of POSS, DPOSS, for large-scale subset selection and the PONSS algorithm for noisy subset selection, respectively; finally introduces the large-scale noisy subset selection problem.

A. Subset Selection

Let $\mathbb{R}$ and $\mathbb{R}^+$ denote the set of reals and non-negative reals, respectively. Let $V = \{v_1, v_2, \ldots, v_n\}$ be a ground set. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. In this paper, we study the set functions $f : 2^V \to \mathbb{R}$. In other words, a solution is a subset of V. Definition 1 gives the general subset selection problem, which is to choose a subset S from a given ground set V that maximizes some given objective function f and meanwhile satisfies the size constraint $|S| \leq k$.

Definition 1 (Subset Selection)**.** *Given a ground set $V = \{v_1, v_2, \ldots, v_n\}$, a set function $f : 2^V \to \mathbb{R}$, and a budget $k \in [n]$, to find a subset of V with at most k items that maximizes f, that is,*

$$\arg\max_{S \subseteq V} f(S) \quad s.t. \quad |S| \leq k. \tag{4}$$

Next, we introduce two properties of set functions, i.e., monotonicity and submodularity, that will be used in this paper. The monotonicity of a set function is shown in Definition 2, which means that as a set extends, the corresponding function value increases. Note that monotone here is assumed to be monotone non-decreasing. In this paper, monotone functions are assumed without loss of generality (w.l.o.g.) to be normalized, i.e., $f(\emptyset) = 0$.

Definition 2 (Monotone)**.** *A set function $f : 2^V \to \mathbb{R}$ is monotone if it holds that*

$$\forall S \subseteq T \subseteq V : f(S) \leq f(T).$$

The submodularity of a set function is shown in Definition 3, which means that the diminishing returns property is satisfied, i.e., as a set extends, the increment on the function brought by adding a same item decreases. The submodularity has several equivalent definitions [30], e.g.,

$$\forall S \subseteq T \subseteq V : \quad f(T) - f(S) \leq \sum\nolimits_{v \in T \setminus S} \big(f(S \cup \{v\}) - f(S)\big), \tag{5}$$

which will also be used in our analysis.

Definition 3 (Submodular)**.** *A set function $f : 2^V \to \mathbb{R}$ is submodular if it holds that*

$$\forall S \subseteq T \subseteq V, v \notin T : \quad f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T). \tag{6}$$

For a general set function f, which is not necessarily submodular, several concepts of *approximate submodularity* have been proposed to measure the closeness of f to submodularity. Here, we give two submodularity ratios in Definitions 4 and 5, which will be used in our analysis. The γ-submodularity ratio is defined based on Eq. (5), whereas the α-submodularity ratio is based on Eq. (6). Note that $\gamma_{S,l}(f)$ and $\alpha(f)$ will be simplified as $\gamma_{S,l}$ and α, when the meaning of f is clear.

Definition 4 (γ-Submodularity Ratio [6])**.** *For a set function $f : 2^V \to \mathbb{R}$, the γ-submodularity ratio with respect to a set $S \subseteq V$ and a parameter $l \geq 1$ is defined as*

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

Definition 5 (α-Submodularity Ratio [43])**.** *For a set function $f : 2^V \to \mathbb{R}$, the α-submodularity ratio is defined as*

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

Since monotone set functions are considered in this paper, we make the observations in Remark 1. Note that the direct computation of these submodularity ratios is very difficult, which requires exponential time. To use them, we need to derive some lower bounds, which are easy to be calculated. In [3], [6], [12], [35], lower bounds on $\gamma_{S,l}(f)$ and $\alpha(f)$ have been derived for some concrete applications of subset selection where the objective functions can be non-submodular, such as Bayesian experimental design, non-parametric learning and sparse regression.

Remark 1. *For a monotone set function $f : 2^V \to \mathbb{R}^+$, it holds that*

1) $0 \leq \alpha(f) \leq 1$ *and* $\forall S, l : 0 \leq \gamma_{S,l}(f) \leq 1$*;*
2) *f is submodular iff $\alpha(f) = 1$;*
3) *f is submodular iff $\forall S, l : \gamma_{S,l}(f) = 1$.*

Here is one application of subset selection, i.e., sparse regression, that will be used in our experiments. As shown in Definition 6, the goal is to select a limited number of observation variables to best approximate a target variable by linear regression. Its objective function $R^2_{z,S}$ has been proved to be monotone, but not necessarily submodular [6]. Note that when we define the mean squared error $\mathrm{MSE}_{z,S}$, S and its index set $\{i \mid v_i \in S\}$ are not distinguished for notational convenience.

Definition 6 (Sparse Regression [26])**.** *Given a ground set $V = \{v_1, v_2, \ldots, v_n\}$ containing n observation variables, a variable z to be predicted, and a budget $k \in [n]$, to find a subset of V with at most k observation variables that maximizes the objective,* squared multiple correlation $R^2_{z,S}$ *[9], [21], that is,*

$$\arg\max_{S \subseteq V} \left(R^2_{z,S} = \frac{\mathrm{Var}(z) - \mathrm{MSE}_{z,S}}{\mathrm{Var}(z)}\right) \quad s.t. \quad |S| \leq k,$$

where $\mathrm{Var}(z)$ denotes the variance of z, and $\mathrm{MSE}_{z,S}$ denotes the mean squared error *of using S to predict z by linear regression, i.e.,*

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

For solving the subset selection problem, which is NP-hard, the greedy algorithm has been shown to be a good approximation solver. The procedure of the greedy algorithm is simple. Starting from the empty set, it continues to greedily add one item with the current largest marginal gain on f, until k items have been selected. Let S denote the subset returned by the greedy algorithm. It has been proved that the greedy algorithm can obtain an approximation ratio of $(1-e^{-\gamma_{S,k}})$ [7] for the subset selection problem with monotone objective functions. For the application of sparse regression where f can be non-submodular, this ratio reaches the best known one [7]. For the problems with submodular f, this ratio is specialized to be $(1-1/e)$, which is optimal [29], [30]. Note that by Remark 1, it is known that when f is submodular, $\gamma_{S,k}=1$.

[The POSS algorithm] Recently, Qian et al. [36] proposed an anytime randomized algorithm for subset selection, POSS, by using the idea of multi-objective evolutionary optimization. The main idea of POSS is to transform the original subset selection problem as a bi-objective optimization problem, then solve the problem by a MOEA, and finally output the best feasible solution from the generated non-dominated solution set. Note that for a constrained optimization problem, a solution is (in)feasible if it does (not) satisfy the constraints.

In the following, the POSS algorithm will be introduced in detail. A subset S of V can be naturally represented by a Boolean vector $\boldsymbol{s}\in\{0,1\}^n$, where $s_i=1$ if $v_i\in S$ and $s_i=0$ otherwise. In other words, for a Boolean vector $\boldsymbol{s}\in\{0,1\}^n$, the corresponding subset is $\{v_i \mid s_i=1\}$. In this paper, a Boolean vector $\boldsymbol{s}\in\{0,1\}^n$ and its corresponding subset will not be distinguished for notational convenience.

The POSS algorithm transforms the subset selection problem, i.e., Eq. (4), as a bi-objective optimization problem: maximizing the objective function f and meanwhile minimizing the subset size $|\boldsymbol{s}|$. That is,

$$\arg\min_{\boldsymbol{s}\in\{0,1\}^n} \quad (f_1(\boldsymbol{s}), f_2(\boldsymbol{s})), \tag{7}$$
$$\text{where} \quad f_1(\boldsymbol{s})=\begin{cases}+\infty, & |\boldsymbol{s}|\geq 2k\\ -f(\boldsymbol{s}), & \text{otherwise}\end{cases}, \quad f_2(\boldsymbol{s})=|\boldsymbol{s}|.$$

Note that minimizing f_1 is equivalent to maximizing f. The goal of setting $f_1(\boldsymbol{s})=+\infty$ for $|\boldsymbol{s}|\geq 2k$ is to exclude overly infeasible solutions, which might be useless and will decrease the efficiency of the optimization process.

To solve this transformed bi-objective optimization problem, POSS employs a simple MOEA, i.e., lines 1-9 of Algorithm 1. This employed MOEA is inspired from the GSEMO algorithm [17], [24], [25], [37], widely used in the theoretical analyses of MOEAs. Starting from the special solution 0^n which represents the empty set (line 1), it iteratively uses bit-wise mutation and domination-based comparison to improve the solutions maintained in the population P (lines 2-9). In each iteration, a parent solution is first selected from P uniformly at random (line 3); then an offspring solution $\boldsymbol{s}'$ is generated by applying the bit-wise mutation operator to $\boldsymbol{s}$ (line 4); finally the generated offspring solution $\boldsymbol{s}'$ is used to update the population P according to the domination-based comparison (lines 5-7).

Algorithm 1 POSS Algorithm [36]

Input: a ground set $V=\{v_1,v_2,\ldots,v_n\}$, an objective function $f:2^V\to\mathbb{R}$, a budget $k\in[n]$
Parameter: the number T of iterations
Output: a subset S of V with $|S|\leq k$
Process:

1: Let $\boldsymbol{s}=0^n$, $P=\{\boldsymbol{s}\}$, and let $t=0$;
2: **while** $t<T$ **do**
3: Select $\boldsymbol{s}$ from P uniformly at random;
4: Generate $\boldsymbol{s}'$ by applying bit-wise mutation to $\boldsymbol{s}$;
5: **if** $\nexists \boldsymbol{z}\in P$ such that $\boldsymbol{z}\prec\boldsymbol{s}'$ **then**
6: $P=(P\setminus\{\boldsymbol{z}\in P \mid \boldsymbol{s}'\preceq\boldsymbol{z}\})\cup\{\boldsymbol{s}'\}$
7: **end if**
8: $t=t+1$
9: **end while**
10: **return** $\arg\max_{\boldsymbol{s}\in P,|\boldsymbol{s}|\leq k} f(\boldsymbol{s})$

The bit-wise mutation operator is shown in Definition 7. It is a global search operator. The probability of generating $\boldsymbol{s}'$ from $\boldsymbol{s}$ is $(1/n)^{H(\boldsymbol{s},\boldsymbol{s}')}(1-1/n)^{n-H(\boldsymbol{s},\boldsymbol{s}')}$, where $H(\boldsymbol{s},\boldsymbol{s}')$ denotes the Hamming distance between two Boolean vectors $\boldsymbol{s}$ and $\boldsymbol{s}'$.

Definition 7 (Bit-wise Mutation)**.** *For a solution $\boldsymbol{s}\in\{0,1\}^n$, the bit-wise mutation operator generates a new offspring solution by flipping each bit of $\boldsymbol{s}$ with probability $1/n$.*

In multi-objective optimization, the domination relationship is often used to compare two solutions. Here, we only give the case of bi-objective minimization in Definition 8 for simplicity. When using the domination-based comparison to update the population P in lines 5-7 of Algorithm 1, the offspring solution $\boldsymbol{s}'$ will be included into P if no solution in P dominates it, and meanwhile those solutions in P weakly dominated by $\boldsymbol{s}'$ will be deleted. Thus, the population P will always contain the non-dominated solutions generated so-far.

Definition 8 (Domination)**.** *Given a bi-objective minimization problem $\min(f_1(\boldsymbol{s}),f_2(\boldsymbol{s}))$. For two solutions $\boldsymbol{s}$ and $\boldsymbol{s}'$,*

1) $\boldsymbol{s}$ weakly dominates $\boldsymbol{s}'$*, denoted as $\boldsymbol{s}\preceq\boldsymbol{s}'$, if*
$$f_1(\boldsymbol{s})\leq f_1(\boldsymbol{s}')\wedge f_2(\boldsymbol{s})\leq f_2(\boldsymbol{s}');$$
2) $\boldsymbol{s}$ dominates $\boldsymbol{s}'$*, denoted as $\boldsymbol{s}\prec\boldsymbol{s}'$, if*
$$\boldsymbol{s}\preceq\boldsymbol{s}'\wedge(f_1(\boldsymbol{s})<f_1(\boldsymbol{s}')\vee f_2(\boldsymbol{s})<f_2(\boldsymbol{s}'));$$
3) $\boldsymbol{s},\boldsymbol{s}'$ *are* incomparable *if neither $\boldsymbol{s}\preceq\boldsymbol{s}'$ nor $\boldsymbol{s}'\preceq\boldsymbol{s}$.*

When the algorithm terminates after T iterations, we get a set P of non-dominated solutions. Since the original problem Eq. (4) is a constrained optimization problem, the best feasible solution in P, i.e., the solution with the smallest f_1 value (or equivalently the largest f value) while satisfying the size constraint in P, is output as the final solution (line 10).

It has been shown that POSS can be better than the greedy algorithm [36]. In theory, POSS with $\mathbb{E}[T]\leq 2ek^2n$ can obtain the same general approximation guarantee as the greedy algorithm, for the subset selection problem with monotone objective functions. Note that $\mathbb{E}[\cdot]$ denotes the expectation

Algorithm 2 DPOSS Algorithm [33]

Input: a ground set $V = \{v_1, v_2, \ldots, v_n\}$, an objective function $f : 2^V \to \mathbb{R}$, a budget $k \in [n]$, the number m of machines

Parameter: the number of iterations at each machine, i.e., $T_1, T_2, \ldots, T_m, T_{m+1}$

Output: a subset S of V with $|S| \leq k$

Process:

1: Partition V into m sets $V_1, V_2, \ldots, V_m$ arbitrarily such that each V_i can fit on one single 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** $\arg\max_{\boldsymbol{s} \in \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{m+1}\}} f(\boldsymbol{s})$

of a random variable. In experiments, POSS can achieve significantly better performance than the greedy algorithm on the application of sparse regression.

B. Large-scale Subset Selection

As the advance of data collection and storage, the applications of subset selection often meet with massive data, which can be too large to be stored on one single machine. Although with excellent performance, POSS requires centralized access to the whole data set V, because subsets of V randomly generated by bit-wise mutation need to be evaluated on one machine. Thus, POSS does not apply to large-scale subset selection tasks.

In [33], Qian et al. thus proposed a distributed version of POSS, DPOSS, as described in Algorithm 2. DPOSS uses the divide and conquer idea, and is implemented in two rounds. For the first round, i.e., lines 1-2 of Algorithm 2, the ground set V is partitioned into $V_1, V_2, \ldots, V_m$ where $V = \cup_{i=1}^{m} V_i \ \wedge \ \forall i \neq j : V_i \cap V_j = \emptyset$; these m sets are distributed over m machines; then POSS is run on each machine in parallel, producing m subsets $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_m$. For the second round, i.e., lines 3-4 of Algorithm 2, POSS is run on the union $\cup_{i=1}^{m} \boldsymbol{s}_i$ of these m subsets generated in the first round, resulting in another subset $\boldsymbol{s}_{m+1}$. DPOSS finally returns the best one among these $(m+1)$ subsets, i.e., line 5.

For the applications of maximum coverage and sparse regression, DPOSS has empirically shown to achieve close performance to the centralized POSS algorithm, and achieve superior performance than the state-of-the-art distributed greedy algorithm, RANDGREEDI [1], [27]. It has also been proved that for the subset selection problem with monotone objective functions, the approximation performance of DPOSS is bounded, as shown in Theorem 1. Note that for an arbitrary partition $\{V_1, V_2, \ldots, V_m\}$ of V, we use notations as follows: $n_i = |V_i|$ and $n_{\max} = \max\{n_i \mid i \in [m]\}$.

Theorem 1 (Theorem 1 of [33])**.** *For the subset selection problem where the objective function f is monotone, using $\mathbb{E}[\max\{T_i \mid i \in [m]\}] = O(k^2 n_{\max}(1 + \log m))$, DPOSS finds a subset $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq k$ and*

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

where $\gamma_{\min} = \min_{\boldsymbol{s} \subseteq V : |\boldsymbol{s}| = k-1} \gamma_{\boldsymbol{s},k}$, and OPT denotes the optimal function value.

C. Noisy Subset Selection

In real applications of subset selection such as influence maximization [22] and sparse regression [26], we can often obtain only a noisy objective function value $F(\boldsymbol{s})$, rather than the exact one $f(\boldsymbol{s})$. For example, computing the objective $R^2_{z,S}$ of sparse regression is time-consuming if using all instances, and thus, a random sample of instances is often used to estimate $R^2_{z,S}$, which, however, introduces noise. Two commonly used noise models [16], [20] are shown as follows.

Definition 9 (Multiplicative Noise)**.** *Let f and F denote the true and noisy objective functions, respectively. It holds that*

$$(1 - \epsilon) \cdot f(\boldsymbol{s}) \leq F(\boldsymbol{s}) \leq (1 + \epsilon) \cdot f(\boldsymbol{s}), \tag{8}$$

where $0 \leq \epsilon \leq 1$.

Definition 10 (Additive Noise)**.** *Let f and F denote the true and noisy objective functions, respectively. It holds that*

$$f(\boldsymbol{s}) - \epsilon \leq F(\boldsymbol{s}) \leq f(\boldsymbol{s}) + \epsilon, \tag{9}$$

where $\epsilon \geq 0$.

The presence of noise can degrade the performance of POSS. Qian et al. [34] thus proposed a noise-tolerant version of POSS, PONSS, as described in Algorithm 3. The main difference is that POSS uses the domination relationship in Definition 8 for comparing two solutions, whereas PONSS uses a conservative strategy, θ-domination, which intuitively means that a solution better than the other requires its objective value to be larger by at least a threshold. θ-domination is inspired by the noise-handling strategy, threshold selection [2], [38]. For the two noise models, multiplicative and additive, the corresponding θ-dominations are shown in Definitions 11 and 12, respectively. Note that these two definitions are specialized to the bi-objective minimization problem Eq. (7), where $f_1(\boldsymbol{s}) = -f(\boldsymbol{s})$. Thus, $f_1(\boldsymbol{s}) \leq \frac{1+\theta}{1-\theta} \cdot f_1(\boldsymbol{s}')$ is equivalent to $f(\boldsymbol{s}) \geq \frac{1+\theta}{1-\theta} \cdot f(\boldsymbol{s}')$, and $f_1(\boldsymbol{s}) \leq f_1(\boldsymbol{s}') - 2\theta$ is equivalent to $f(\boldsymbol{s}) \geq f(\boldsymbol{s}') + 2\theta$.

Definition 11 (Multiplicative θ-Domination)**.** *Given a bi-objective minimization problem $\min(f_1(\boldsymbol{s}), f_2(\boldsymbol{s}))$ and a parameter $\theta \in [0, 1]$. For two solutions $\boldsymbol{s}$ and $\boldsymbol{s}'$,*

1) *$\boldsymbol{s}$ weakly dominates $\boldsymbol{s}'$, denoted as $\boldsymbol{s} \preceq_\theta \boldsymbol{s}'$, if*

$$f_1(\boldsymbol{s}) \leq \frac{1+\theta}{1-\theta} \cdot f_1(\boldsymbol{s}') \wedge f_2(\boldsymbol{s}) \leq f_2(\boldsymbol{s}');$$

2) *$\boldsymbol{s}$ dominates $\boldsymbol{s}'$, denoted as $\boldsymbol{s} \prec_\theta \boldsymbol{s}'$, if*

$$\boldsymbol{s} \preceq_\theta \boldsymbol{s}' \wedge \left(f_1(\boldsymbol{s}) < \frac{1+\theta}{1-\theta} \cdot f_1(\boldsymbol{s}') \vee f_2(\boldsymbol{s}) < f_2(\boldsymbol{s}')\right);$$

3) *$\boldsymbol{s}, \boldsymbol{s}'$ are* incomparable *if neither $\boldsymbol{s} \preceq_\theta \boldsymbol{s}'$ nor $\boldsymbol{s}' \preceq_\theta \boldsymbol{s}$.*

Definition 12 (Additive θ-Domination)**.** *Given a bi-objective minimization problem $\min(f_1(\boldsymbol{s}), f_2(\boldsymbol{s}))$ and a parameter $\theta \geq 0$. For two solutions $\boldsymbol{s}$ and $\boldsymbol{s}'$,*

1) *$\boldsymbol{s}$ weakly dominates $\boldsymbol{s}'$, denoted as $\boldsymbol{s} \preceq_\theta \boldsymbol{s}'$, if*

$$f_1(\boldsymbol{s}) \leq f_1(\boldsymbol{s}') - 2\theta \wedge f_2(\boldsymbol{s}) \leq f_2(\boldsymbol{s}');$$

Algorithm 3 PONSS Algorithm [34]

Input: a ground set $V = \{v_1, v_2, \ldots, v_n\}$, a noisy objective function $F : 2^V \to \mathbb{R}$, a budget $k \in [n]$
Parameter: T, θ, B
Output: a subset S of V with $|S| \leq k$
Process:

1: Let $\boldsymbol{s} = 0^n$, $P = \{\boldsymbol{s}\}$, and let $t = 0$;
2: **while** $t < T$ **do**
3: Select $\boldsymbol{s}$ from P uniformly at random;
4: Generate $\boldsymbol{s}'$ by applying bit-wise mutation to $\boldsymbol{s}$;
5: **if** $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec_\theta \boldsymbol{s}'$ **then**
6: $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{s}' \preceq_\theta \boldsymbol{z}\}) \cup \{\boldsymbol{s}'\}$;
7: $Q = \{\boldsymbol{z} \in P \mid |\boldsymbol{z}| = |\boldsymbol{s}'|\}$;
8: **if** $|Q| = B + 1$ **then**
9: $P = P \setminus Q$ and let $j = 0$;
10: **while** $j < B$ **do**
11: Select two solutions $\boldsymbol{z}_1, \boldsymbol{z}_2$ from Q uniformly at random without replacement;
12: Evaluate $F(\boldsymbol{z}_1)$ and $F(\boldsymbol{z}_2)$;
13: Let $\hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{z} \in \{\boldsymbol{z}_1, \boldsymbol{z}_2\}} F(\boldsymbol{z})$ (breaking ties randomly);
14: $P = P \cup \{\hat{\boldsymbol{z}}\}$, $Q = Q \setminus \{\hat{\boldsymbol{z}}\}$, and $j = j + 1$
15: **end while**
16: **end if**
17: **end if**
18: $t = t + 1$
19: **end while**
20: **return** $\arg\max_{\boldsymbol{s} \in P, |\boldsymbol{s}| \leq k} F(\boldsymbol{s})$

2) $\boldsymbol{s}$ dominates $\boldsymbol{s}'$, *denoted as* $\boldsymbol{s} \prec_\theta \boldsymbol{s}'$, *if*

$$\boldsymbol{s} \preceq_\theta \boldsymbol{s}' \wedge (f_1(\boldsymbol{s}) < f_1(\boldsymbol{s}') - 2\theta \vee f_2(\boldsymbol{s}) < f_2(\boldsymbol{s}'))\,;$$

3) $\boldsymbol{s}, \boldsymbol{s}'$ *are* incomparable *if neither* $\boldsymbol{s} \preceq_\theta \boldsymbol{s}'$ *nor* $\boldsymbol{s}' \preceq_\theta \boldsymbol{s}$.

For each subset size, if using domination, only the solution with the best noisy F value is maintained in the population P; if using θ-domination, solutions with close F values will all be maintained in P, reducing the risk of losing good solutions. However, the population size, i.e., $|P|$, can become very large by using θ-domination. Thus, PONSS introduces an extra procedure to control the size of P in lines 7-16 of Algorithm 3. For each subset size i, the number of solutions in P is limited by at most B, which is a parameter. When the number exceeds B, which actually must be $B + 1$, one solution with size i will be removed as follows: two solutions with size i are randomly selected without replacement and the better one is kept; after repeating this process B times, the remaining solution with size i is removed.

In [34], it has been proved that under either multiplicative or additive noise, PONSS with $\theta \geq \epsilon$ can achieve an approximation guarantee significantly better than POSS, in polynomial time. The superior performance of PONSS has also been empirically shown on the two noisy applications, influence maximization and sparse regression.

D. Large-scale Noisy Subset Selection

When large-scale subset selection meets with noise, the problem of large-scale noisy subset selection emerges. In fact, even when the objective function evaluation is originally exact, noise can be produced due to the distribution. For example, for the application of unsupervised feature selection [13] where each item is a column vector, the objective function $\|V - SS^+V\|_F^2$ relies on the whole data set V, but can only be estimated using the local data allocated to each machine after the distribution, which thus produces noise.

Although DPOSS has shown excellent performance for large-scale subset selection, its performance under noise is not yet known. PONSS performs well for noisy subset selection, but does not apply to large-scale applications. In the following, we first examine whether DPOSS can still achieve good performance for large-scale noisy subset selection.

III. Theoretical Analysis of DPOSS under Noise

In this section, we theoretically analyze the performance of DPOSS under noise. For subset selection with monotone objective functions, we prove the approximation guarantee of DPOSS in Theorem 2 under multiplicative noise. The proof idea is inspired by that of Theorem 1 in [33], which gives the approximation guarantee of DPOSS without noise. In the proof, we will use Lemmas 1 and 2. Lemma 1 shows that the objective f value of the best subset over all machines can be lower bounded. Lemma 2 gives the recursive relation of the noisy objective F value between any set and its resulting superset by adding a specific item.

Lemma 1 (Lemma 1 of [33])**.** *For any partition* $\{V_1, \ldots, V_m\}$ *of* V*, let* $\boldsymbol{o}_i$ *denote an optimal subset of* V_i*, i.e.,* $\boldsymbol{o}_i \in \arg\max_{\boldsymbol{s} \subseteq V_i : |\boldsymbol{s}| \leq k} f(\boldsymbol{s})$*, where* $i \in [m]$*. It holds that*

$$\max\{f(\boldsymbol{o}_i) \mid i \in [m]\} \geq \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT}.$$

Lemma 2 (Lemma 2 of [34])**.** *For any* $\boldsymbol{s} \subseteq V_i$*, where* $i \in [m]$*, there exists one item* $v \in V_i \setminus \boldsymbol{s}$ *satisfying that*

$$F(\boldsymbol{s} \cup \{v\}) \geq \left(\frac{1-\epsilon}{1+\epsilon}\right)\left(1 - \frac{\gamma_{\boldsymbol{s},k}}{k}\right) F(\boldsymbol{s}) + \frac{(1-\epsilon)\gamma_{\boldsymbol{s},k}}{k} f(\boldsymbol{o}_i).$$

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

$$f(\boldsymbol{s}) \geq \frac{1}{1 + \frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}} \cdot \left(1 - \left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}\right) \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT},$$

where $\gamma_{\min} = \min_{\boldsymbol{s} \subseteq V : |\boldsymbol{s}| = k-1} \gamma_{\boldsymbol{s},k}$*, and* OPT *denotes the optimal function value.*

Proof. We consider the first round of DPOSS, where POSS is run on each V_i to find a subset $\boldsymbol{s}_i$. To reach the desired approximation guarantee, we only need to analyze $\max\{T_i \mid$

$i \in [m]\}$, i.e., the maximal number of iterations of POSS at each machine, until

$$\forall i \in [m]: \ F(\boldsymbol{s}_i) \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k\left(1-\frac{\gamma_{\min}}{k}\right)^k\right) \cdot f(\boldsymbol{o}_i). \tag{10}$$

Note that the selection in line 5 of DPOSS, i.e., Algorithm 2, relies on the noisy objective F value now. Thus, for the F value of the returned subset, denoted as $\boldsymbol{s}^*$, it holds that

$$\begin{aligned} F(\boldsymbol{s}^*) &= \max\{F(\boldsymbol{s}_i) \mid i \in [m+1]\} \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \\ &\cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k\left(1-\frac{\gamma_{\min}}{k}\right)^k\right) \cdot \max\{f(\boldsymbol{o}_i) \mid i \in [m]\} \\ &\geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}\right) \\ &\quad \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT}, \end{aligned} \tag{11}$$

where the first inequality holds by Eq. (10), and the last one holds by $(1-\gamma_{\min}/k)^k \leq e^{-\gamma_{\min}}$ and Lemma 1. According to the definition of multiplicative noise, i.e., Definition 9, we have $F(\boldsymbol{s}^*) \leq (1+\epsilon) \cdot f(\boldsymbol{s}^*)$. Thus,

$$\begin{aligned} f(\boldsymbol{s}^*) &\geq \frac{\frac{1-\epsilon}{1+\epsilon}\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}\right) \\ &\quad \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT} \\ &= \frac{1}{1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}\right) \\ &\quad \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT}, \end{aligned}$$

implying that the desired approximation guarantee is obtained.

Thus, we only need to examine $\max\{T_i \mid i \in [m]\}$ until Eq. (10) holds. For POSS running on V_i, where $i \in [m]$, we use $J^i_{\max}$ to denote the maximal value of $j \in \{0,1,\ldots,k\}$, satisfying that the population P contains a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq j$ and $F(\boldsymbol{s}) \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot (1-(\frac{1-\epsilon}{1+\epsilon})^j(1-\frac{\gamma_{\min}}{k})^j) \cdot f(\boldsymbol{o}_i)$. In other words,

$$J^i_{\max} = \max\{j \in \{0,1,\ldots,k\} \mid \exists \boldsymbol{s} \in P, |\boldsymbol{s}| \leq j \wedge F(\boldsymbol{s}) \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^j\left(1-\frac{\gamma_{\min}}{k}\right)^j\right) \cdot f(\boldsymbol{o}_i)\}.$$

It is clear that $J^i_{\max} = k$ implies that P contains a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq k$ and $F(\boldsymbol{s}) \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot (1-(\frac{1-\epsilon}{1+\epsilon})^k(1-\frac{\gamma_{\min}}{k})^k) \cdot f(\boldsymbol{o}_i)$, and thus the subset $\boldsymbol{s}_i$ returned by POSS running on V_i satisfies that $F(\boldsymbol{s}_i) \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot (1-(\frac{1-\epsilon}{1+\epsilon})^k(1-\frac{\gamma_{\min}}{k})^k) \cdot f(\boldsymbol{o}_i)$. To make Eq. (10) holds, it is sufficient to make $\min\{J^i_{\max} \mid i \in [m]\} = k$. Thus, our focus is now on analyzing $\max\{T_i \mid i \in [m]\}$ until $\min\{J^i_{\max} \mid i \in [m]\} = k$.

As the initial solution of POSS is 0^n, $\forall i \in [m]: J^i_{\max} = 0$, implying that $\min\{J^i_{\max} \mid i \in [m]\}$ is initially 0. We assume w.l.o.g. that $\min\{J^i_{\max} \mid i \in [m]\} = j < k$ and $|\{i \in [m] \mid J^i_{\max} = j\}| = l$, where $l \in [m]$. Let $\boldsymbol{z}_i$ denote a solution corresponding to $J^i_{\max}$, implying that $|\boldsymbol{z}_i| \leq J^i_{\max}$ and

$$F(\boldsymbol{z}_i) \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}(1-\frac{\gamma_{\min}}{k})} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^{J^i_{\max}}\left(1-\frac{\gamma_{\min}}{k}\right)^{J^i_{\max}}\right) \cdot f(\boldsymbol{o}_i). \tag{12}$$

It is clear that $J^i_{\max}$ does not decrease. If $\boldsymbol{z}_i$ is kept in the population P, the claim obviously holds. If $\boldsymbol{z}_i$ is deleted from P in line 6 of Algorithm 1, it must hold that the newly generated solution $\boldsymbol{s}' \preceq \boldsymbol{z}_i$, implying that $|\boldsymbol{s}'| \leq |\boldsymbol{z}_i| \wedge F(\boldsymbol{s}') \geq F(\boldsymbol{z}_i)$. According to the fact that $J^i_{\max}$ does not decrease for each i, we conclude that j does not decrease and the corresponding l does not increase.

Next, we analyze the probability of increasing $J^i_{\max}$ in one iteration for $i \in [m]$. Lemma 2 shows that $\forall i \in [m]$, including one specific item into $\boldsymbol{z}_i$, i.e., flipping one specific 0 bit of $\boldsymbol{z}_i$, can generate an offspring solution $\boldsymbol{s}'$, satisfying that

$$\begin{aligned} F(\boldsymbol{s}') &\geq \left(\frac{1-\epsilon}{1+\epsilon}\right)\left(1-\frac{\gamma_{\boldsymbol{z}_i,k}}{k}\right)F(\boldsymbol{z}_i) + \frac{(1-\epsilon)\gamma_{\boldsymbol{z}_i,k}}{k}f(\boldsymbol{o}_i) \\ &= \frac{1-\epsilon}{1+\epsilon}F(\boldsymbol{z_i}) + \left(f(\boldsymbol{o}_i) - \frac{F(\boldsymbol{z}_i)}{1+\epsilon}\right)\frac{(1-\epsilon)\gamma_{\boldsymbol{z}_i,k}}{k}. \end{aligned}$$

Consider $J^i_{\max} < k$. By the definition of the γ-submodularity ratio, i.e., Definition 4, it can be verified that $\gamma_{\boldsymbol{s},k}$ decreases with $\boldsymbol{s}$. As $|\boldsymbol{z}_i| \leq J^i_{\max} < k$ and $\gamma_{\min} = \min_{\boldsymbol{s} \subseteq V:|\boldsymbol{s}|=k-1}\gamma_{\boldsymbol{s},k}$, we have $\gamma_{\boldsymbol{z}_i,k} \geq \gamma_{\min}$. Because $f(\boldsymbol{o}_i) - \frac{F(\boldsymbol{z}_i)}{1+\epsilon} \geq f(\boldsymbol{z}_i) - \frac{F(\boldsymbol{z}_i)}{1+\epsilon} \geq 0$, we have

$$F(\boldsymbol{s}') \geq \left(\frac{1-\epsilon}{1+\epsilon}\right)\left(1-\frac{\gamma_{\min}}{k}\right)F(\boldsymbol{z}_i) + \frac{(1-\epsilon)\gamma_{\min}}{k} \cdot f(\boldsymbol{o}_i).$$

By applying Eq. (12) to the above inequality, we have

$$F(\boldsymbol{s}') \geq \frac{(1-\epsilon)\frac{\gamma_{\min}}{k}}{1-\frac{1-\epsilon}{1+\epsilon}\left(1-\frac{\gamma_{\min}}{k}\right)} \cdot \left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^{J^i_{\max}+1}\left(1-\frac{\gamma_{\min}}{k}\right)^{J^i_{\max}+1}\right) \cdot f(\boldsymbol{o}_i).$$

Considering $|\boldsymbol{s}'| = |\boldsymbol{z}_i| + 1 \leq J^i_{\max} + 1$, $\boldsymbol{s}'$ will enter into P. Otherwise, according to line 5 of Algorithm 1, we have $\exists \boldsymbol{z} \in P : \boldsymbol{z} \prec \boldsymbol{s}'$, contradicting with the definition of $J^i_{\max}$. After adding $\boldsymbol{s}'$ into P, $J^i_{\max}$ will increase by at least 1. Thus, $J^i_{\max}$ can increase by at least 1 through flipping a specific 0-bit of $\boldsymbol{z}_i$. In line 3 of Algorithm 1, $\boldsymbol{z}_i$ is selected with probability $(1/|P|)$ due to uniform selection. In line 4, i.e., bit-wise mutation, only a specific bit of $\boldsymbol{z}_i$ is flipped with probability $(1/n_i)(1-1/n_i)^{n_i-1}$, where n_i is the size of V_i. Thus, the probability of increasing $J^i_{\max}$ by at least 1 in one iteration is at least $(1/|P|) \cdot (1/n_i)(1-1/n_i)^{n_i-1} \geq 1/(en_i|P|)$. Due to the domination-based comparison in lines 5-7 of Algorithm 1, the population P must contain incomparable solutions, implying that there is at most one corresponding solution in P for each possible value of some objective. As solutions with size at least $2k$ are excluded from the population P, we have $|P| \leq 2k$. Thus, the probability of increasing $J^i_{\max}$ by at least 1 in one iteration is at least $1/(2ekn_i)$.

Now, we can analyze the expected number of iterations required to increase j, or equivalently to decrease the corresponding l to 0. To decrease l by at least 1, it is sufficient to increase at least one $J^i_{\max}$ with value j, which happens with probability at least

$$1-\prod_{i:J^i_{\max}=j}\left(1-\frac{1}{2ekn_i}\right)\geq 1-\left(1-\frac{1}{2ekn_{\max}}\right)^l$$

in one iteration. The inequality holds by $n_{\max}=\max\{n_i \mid i\in[m]\}$ and $|\{i\in[m]\mid J^i_{\max}=j\}|=l$. Thus, decreasing l to 0, i.e., increasing j by at least 1, requires at most

$$\begin{aligned}&\sum_{l=1}^{m}\frac{1}{1-\left(1-\frac{1}{2ekn_{\max}}\right)^l}=\sum_{l=1}^{m}1+\frac{1}{\frac{1}{\left(1-\frac{1}{2ekn_{\max}}\right)^l}-1}\\&=\sum_{l=1}^{m}1+\frac{1}{\left(1+\frac{\frac{1}{2ekn_{\max}}}{1-\frac{1}{2ekn_{\max}}}\right)^l-1}\leq\sum_{l=1}^{m}1+\frac{1}{\frac{l}{2ekn_{\max}}}\\&=\sum_{l=1}^{m}1+\frac{2ekn_{\max}-1}{l}=O(kn_{\max}(1+\log m))\end{aligned}$$

iterations in expectation.

To make $\min\{J^i_{\max}\mid i\in[m]\}=k$, it requires to increase j by at most k times. The expected number of iterations is then at most $O(k^2n_{\max}(1+\log m))$, implying

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

Thus, the theorem holds. □

Theorem 3 shows the approximation bound of DPOSS under additive noise. The proof can be accomplished in the way similar to that of Theorem 2. The only difference is that when comparing the true objective $f(\boldsymbol{s})$ with the noisy objective $F(\boldsymbol{s})$, Eq. (9) is used under additive noise, whereas Eq. (8) is used under multiplicative noise.

Theorem 3. *For the subset selection problem, where the objective function f is monotone, under additive noise, using $\mathbb{E}[\max\{T_i\mid i\in[m]\}]=O(k^2n_{\max}(1+\log m))$, DPOSS finds a subset $\boldsymbol{s}$ with $|\boldsymbol{s}|\leq k$ and*

$$f(\boldsymbol{s})\geq(1-e^{-\gamma_{\min}})\cdot\left(\max\left\{\frac{\alpha}{m},\frac{\gamma_{\emptyset,k}}{k}\right\}\cdot\mathrm{OPT}-\frac{2k\epsilon}{\gamma_{\min}}\right)-\left(1-\frac{\gamma_{\min}}{k}\right)^k\epsilon,$$

where $\gamma_{\min}=\min_{\boldsymbol{s}\subseteq V:|\boldsymbol{s}|=k-1}\gamma_{\boldsymbol{s},k}$, and OPT *denotes the optimal function value.*

To examine the influence of noise on the performance of DPOSS, we compare Theorems 2 and 3 with Theorem 1, and make the following observations.

Remark 2. *For DPOSS solving the subset selection problem with monotone objective functions,*

1) *the approximation guarantee under multiplicative noise in Theorem 2 is worse than that without noise in Theorem 1;*
2) *the approximation guarantee under additive noise in Theorem 3 is worse than that without noise in Theorem 1.*

Algorithm 4 DPONSS Algorithm

Input: a ground set $V=\{v_1,v_2,\ldots,v_n\}$, a noisy objective function $F:2^V\to\mathbb{R}$, a budget $k\in[n]$, the number m of machines
Parameter: $T_1,T_2,\ldots,T_m,T_{m+1}$, θ, B
Output: a subset S of V with $|S|\leq k$
Process:

1: Partition V into m sets $V_1,V_2,\ldots,V_m$ arbitrarily such that each V_i can fit on one single machine;
2: Run PONSS with $(T=T_i,\theta,B)$ 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 PONSS with $(T=T_{m+1},\theta,B)$ 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 for maximization, the approximation guarantee is worse means that it is smaller. The second case in Remark 2 obviously holds, because the approximation bound in Theorem 3 equals that in Theorem 1 minus some additional positive terms. For the first case, it can be verified because

$$\begin{aligned}&\frac{1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}}{1-e^{-\gamma_{\min}}}=\frac{1-\left(1-\frac{2\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}}{1-e^{-\gamma_{\min}}}\\&\leq\frac{1-\left(1-\frac{2k\epsilon}{1+\epsilon}\right)e^{-\gamma_{\min}}}{1-e^{-\gamma_{\min}}}=1+\frac{\frac{2k\epsilon}{1+\epsilon}e^{-\gamma_{\min}}}{1-e^{-\gamma_{\min}}}\\&=1+\frac{\frac{2k\epsilon}{1+\epsilon}}{e^{\gamma_{\min}}-1}\leq 1+\frac{\frac{2k\epsilon}{1+\epsilon}}{\gamma_{\min}}\leq 1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}},\end{aligned}\tag{13}$$

leading to

$$\frac{1}{1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}}\cdot\left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}\right)\leq 1-e^{-\gamma_{\min}}.$$

In fact, we can find that the presence of noise degrades the performance of DPOSS significantly. For example, for the case of f being submodular, where $\alpha=1$ and $\forall\boldsymbol{s},l:\gamma_{\boldsymbol{s},l}=1$, Theorem 1 shows that DPOSS can achieve an approximation guarantee of $\Theta(\max\{1/m,1/k\})$ in the noiseless environment, whereas Theorem 2 shows that under multiplicative noise, DPOSS can achieve such an approximation guarantee only if $\epsilon=O(1/k)$.

IV. The DPONSS Algorithm

In this section, we propose a new distributed algorithm, DPONSS, for large-scale noisy subset selection, and will show that in the noisy environment, DPONSS can achieve an approximation guarantee, much better than DPOSS.

As described in Algorithm 4, DPONSS shares the same structure as DPOSS. The difference is that PONSS, instead of POSS, is run on each machine. It is a two round algorithm. In the first round, the ground set is divided and distributed to several machines, and PONSS is run on each machine in parallel to find a subset. After this, the algorithm enters into the second round, where the subsets generated in the first round are combined on one single machine and PONSS is run again

to find another subset. Finally, the best one is selected from all generated subsets, as the final output.

DPONSS is easy to be implemented in the Hadoop MapReduce framework. The parameters $T_1, T_2, \ldots, T_{m+1}$, θ and B are able to influence the goodness of the output solution, and their relationship will be made clear in the following theoretical analysis.

A. Theoretical Analysis

Theorem 4 shows the approximation guarantee of DPONSS under multiplicative noise. The proof idea is similar to that of Theorem 2. The main difference consists of two aspects: (1) for defining the quantity $J^i_{\max}$, an inductive inequality of f is used, rather than an inductive inequality of F; (2) the number of iterations required to achieve the desired approximation guarantee is analyzed with probability, rather than in expectation. The proof is also inspired by Theorem 5 in [34], which gives the approximation guarantee of PONSS under multiplicative noise. For concise illustration, some existing results in previous analyses will be used directly in the proof of Theorem 4.

To analyze the approximation guarantee of DPONSS, some assumptions are required: (1) the noise is random, i.e., $F(\boldsymbol{s})$ is a random variable; (2) if $f(\boldsymbol{s}) > f(\boldsymbol{s}')$, $F(\boldsymbol{s}) > F(\boldsymbol{s}')$ holds with probability at least $0.5 + \delta$, i.e.,

$$\Pr(F(\boldsymbol{s}) > F(\boldsymbol{s}')) \geq 0.5 + \delta \quad \text{if} \quad f(\boldsymbol{s}) > f(\boldsymbol{s}'), \quad (14)$$

where $0 \leq \delta < 0.5$. Several common noisy settings satisfy this assumption. For example, the i.i.d. noise distribution has been verified in [34].

Theorem 4. *For the subset selection problem, where the objective function f is monotone, under multiplicative noise with the assumption Eq. (14), with probability $(1/2) \cdot (1 - O(k^2 m n_{\max}(\log k + \log m)(1+\log m)/B^{2\delta}))$, DPONSS using $\theta \geq \epsilon$ and $\max\{T_i \mid i \in [m]\} = O(Bk^2 n_{\max}(\log k + \log m)(1 + \log m))$ finds a subset $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq k$ and*

$$f(\boldsymbol{s}) \geq \frac{1-\epsilon}{1+\epsilon} \cdot (1 - e^{-\gamma_{\min}}) \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT},$$

where $\gamma_{\min} = \min_{\boldsymbol{s} \subseteq V : |\boldsymbol{s}| = k-1} \gamma_{\boldsymbol{s},k}$, and OPT denotes the optimal function value.

Proof. The first round of DPONSS is considered, where PONSS is run on each V_i to find a subset $\boldsymbol{s}_i$. To reach the desired approximation guarantee, it is sufficient to make the following equation hold:

$$\forall i \in [m] : F(\boldsymbol{s}_i) \geq (1-\epsilon)\left(1 - \left(1 - \frac{\gamma_{\min}}{k}\right)^k\right) f(\boldsymbol{o}_i). \quad (15)$$

As the analysis of Eq. (11), the F value of the returned subset $\boldsymbol{s}^*$ satisfies that

$$\begin{aligned} F(\boldsymbol{s}^*) &= \max\{F(\boldsymbol{s}_i) \mid i \in [m+1]\} \\ &\geq (1-\epsilon) \cdot \left(1 - \left(1 - \frac{\gamma_{\min}}{k}\right)^k\right) \cdot \max\{f(\boldsymbol{o}_i) \mid i \in [m]\} \\ &\geq (1-\epsilon) \cdot (1 - e^{-\gamma_{\min}}) \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT}. \end{aligned}$$

By the definition of multiplicative noise, we have

$$\begin{aligned} f(\boldsymbol{s}^*) &\geq F(\boldsymbol{s}^*)/(1+\epsilon) \\ &\geq \frac{1-\epsilon}{1+\epsilon} \cdot (1 - e^{-\gamma_{\min}}) \cdot \max\left\{\frac{\alpha}{m}, \frac{\gamma_{\emptyset,k}}{k}\right\} \cdot \mathrm{OPT}, \end{aligned}$$

implying that the desired approximation guarantee is obtained.

For PONSS running on V_i, where $i \in [m]$, we define $J^i_{\max}$ as follows:

$$\begin{aligned} J^i_{\max} = \max\{j \in \{0,1,\ldots,k\} \mid \exists \boldsymbol{s} \in P, |\boldsymbol{s}| \leq j \\ \wedge f(\boldsymbol{s}) \geq \left(1 - \left(1 - \frac{\gamma_{\min}}{k}\right)^j\right) \cdot f(\boldsymbol{o}_i)\}. \end{aligned}$$

To make Eq. (15) hold, it is sufficient to make $\min\{J^i_{\max} \mid i \in [m]\} = k$. $J^i_{\max} = k$ implies that P contains a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq k$ and $f(\boldsymbol{s}) \geq (1 - (1 - \frac{\gamma_{\min}}{k})^k) \cdot f(\boldsymbol{o}_i)$. As the returned subset $\boldsymbol{s}_i$ has the largest noisy objective F value,

$$\begin{aligned} F(\boldsymbol{s}_i) &\geq F(\boldsymbol{s}) \geq (1-\epsilon) \cdot f(\boldsymbol{s}) \\ &\geq (1-\epsilon) \cdot \left(1 - \left(1 - \frac{\gamma_{\min}}{k}\right)^k\right) \cdot f(\boldsymbol{o}_i), \end{aligned}$$

where the second inequality holds due to multiplicative noise, i.e., Definition 9. Thus, we only need to analyze the number of iterations in the first round of DPONSS until $\min\{J^i_{\max} \mid i \in [m]\} = k$.

For any $i \in [m]$, $J^i_{\max}$ is initially 0, because PONSS starts from 0^n. We make the assumption that during the running of PONSS, one solution with the best true objective f value in Q is always retained if lines 8-16 of Algorithm 3 are implemented. From the proof of Theorem 5 in [34], it has been known that $\forall i \in [m]$, $J^i_{\max}$ does not decrease and the probability of increasing $J^i_{\max}$ by at least 1 in one iteration is at least $1/(2eBkn_i) \geq 1/(2eBkn_{\max})$.

It is clear that $\min\{J^i_{\max} \mid i \in [m]\}$ is initially 0. We assume w.l.o.g. that $\min\{J^i_{\max} \mid i \in [m]\} = j < k$ and $|\{i \in [m] \mid J^i_{\max} = j\}| = l$, where $l \in [m]$. In $(2eBkn_{\max}\log(2km)/l)$ iterations of DPONSS, the current l is decreased with probability at least

$$\begin{aligned} 1 - \left(1 - \frac{1}{2eBkn_{\max}}\right)^{l \cdot \frac{2eBkn_{\max}\log(2km)}{l}} &\geq 1 - e^{-\log(2km)} \\ &= 1 - \frac{1}{2km}. \end{aligned}$$

This implies that after running DPONSS for $(\sum_{l=1}^{m} 2eBkn_{\max}\log(2km)/l)$ iterations, the probability of decreasing l to 0, i.e., increasing j by at least 1, is at least

$$\left(1 - \frac{1}{2km}\right)^m \geq 1 - \frac{1}{2k}.$$

To make $\min\{J^i_{\max} \mid i \in [m]\} = k$, it requires to increase j by at most k times. Thus, after

$$\begin{aligned} &k \cdot \sum_{l=1}^{m} \frac{2eBkn_{\max}\log(2km)}{l} \quad (16) \\ &= O(Bk^2 n_{\max}(\log k + \log m)(1 + \log m)) \end{aligned}$$

iterations, $\min\{J^i_{\max} \mid i \in [m]\} = k$ with probability at least

$$\left(1 - \frac{1}{2k}\right)^k \geq \frac{1}{2}. \quad (17)$$

Finally, we need to examine the assumption that during the running of $O(Bk^2 n_{\max}(\log k+\log m)(1+\log m))$ iterations, one solution with the best true objective f value in Q is always retained if lines 8-16 of Algorithm 3 are implemented. From the proof of Theorem 5 in [34], it has been known that when implementing lines 8-16, the solution with the best true objective f value in Q is deleted with probability at most $\frac{4}{e^{1-1/e}B^{1+2\delta}}$. Note that their analysis relies on Eq. (14). Based on the union bound, the probability that the assumption holds is at least

$$\begin{aligned}&1-O(Bk^2 n_{\max}(\log k+\log m)(1+\log m))\cdot m\cdot\frac{4}{e^{1-1/e}B^{1+2\delta}}\\&=1-O(k^2 m n_{\max}(\log k+\log m)(1+\log m)/B^{2\delta}),\end{aligned}\tag{18}$$

where the factor m in the left-hand side corresponds to m machines.

Combining Eqs. (16), (17) and (18) leads to the theorem. □

For DPONSS under additive noise, we prove the approximation guarantee in Theorem 5. The proof is the same as that of Theorem 4, expect that Eq. (9) is used instead of Eq. (8) when comparing $F(\boldsymbol{s})$ with $f(\boldsymbol{s})$.

Theorem 5. *For the subset selection problem, where the objective function f is monotone, under additive noise with the assumption Eq. (14), with probability $(1/2)\cdot(1-O(k^2 m n_{\max}(\log k+\log m)(1+\log m)/B^{2\delta}))$, DPONSS using $\theta\geq\epsilon$ and $\max\{T_i \mid i\in[m]\}=O(Bk^2 n_{\max}(\log k+\log m)(1+\log m))$ finds a subset $\boldsymbol{s}$ with $|\boldsymbol{s}|\leq k$ and*

$$f(\boldsymbol{s})\geq(1-e^{-\gamma_{\min}})\cdot\max\left\{\frac{\alpha}{m},\frac{\gamma_{\emptyset,k}}{k}\right\}\cdot\mathrm{OPT}-2\epsilon,$$

where $\gamma_{\min}=\min_{\boldsymbol{s}\subseteq V:|\boldsymbol{s}|=k-1}\gamma_{\boldsymbol{s},k}$, and OPT denotes the optimal function value.

When the assumption Eq. (14) of noise holds with a constant large δ, using a polynomial B can make the approximation guarantee of DPONSS hold with a constant probability, implying that the maximal number of iterations in the first round of DPONSS is polynomial in expectation. To compare the approximation guarantees of DPONSS and DPOSS under noise, we make the following observations.

Remark 3. *For solving the subset selection problem with monotone objective functions,*

1) *under multiplicative noise, the approximation guarantee of DPONSS in Theorem 4 is better than that of DPOSS in Theorem 2;*
2) *under additive noise, the approximation guarantee of DPONSS in Theorem 5 is better than that of DPOSS in Theorem 3.*

For the first case in Remark 3, it can be verified because

$$\begin{aligned}&\frac{1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}}{\frac{1-\epsilon}{1+\epsilon}(1-e^{-\gamma_{\min}})}\leq\frac{1+\epsilon}{1-\epsilon}+\frac{2k\epsilon}{(1-\epsilon)(e^{\gamma_{\min}}-1)}\\&\quad\text{(the inequality is derived as the analysis of Eq. (13))}\\&=1+\frac{2\epsilon}{1-\epsilon}+\frac{2k\epsilon}{(1-\epsilon)(e^{\gamma_{\min}}-1)}\\&=1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}\left(\frac{\gamma_{\min}}{k}+\frac{\gamma_{\min}}{e^{\gamma_{\min}}-1}\right)\\&\leq 1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}\left(\frac{\gamma_{\min}}{k}+\frac{1}{1+\gamma_{\min}/2}\right)\\&\quad\text{(the inequality holds by } e^{\gamma_{\min}}\geq 1+\gamma_{\min}+\gamma_{\min}^2/2)\\&\leq 1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}\left(\frac{\gamma_{\min}}{3}+\frac{1}{1+\gamma_{\min}/2}\right)\\&\quad\text{(the inequality holds with } k\geq 3)\\&\leq 1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}},\quad\text{(by } \gamma_{\min}\leq 1)\end{aligned}$$

leading to

$$\frac{1}{1+\frac{2k\epsilon}{(1-\epsilon)\gamma_{\min}}}\cdot\left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^k e^{-\gamma_{\min}}\right)\leq\frac{1-\epsilon}{1+\epsilon}(1-e^{-\gamma_{\min}}).$$

The second case can be verified by $\forall k\geq 2:(1-e^{-\gamma_{\min}})\frac{2k\epsilon}{\gamma_{\min}}\geq 2\epsilon$.

In fact, we can find that the approximation guarantee of DPONSS under noise can be significantly better than DPOSS. For example, when the objective function is submodular (where $\alpha=1$ and $\forall\boldsymbol{s},l:\gamma_{\boldsymbol{s},l}=1$) and the noise level ϵ of multiplicative noise is a constant, DPONSS can achieve an approximation ratio of $\Theta(\max\{1/m,1/k\})$, whereas DPOSS only guarantees an approximation ratio of $\Theta((1/k)\cdot\max\{1/m,1/k\})$.

V. Empirical Study

In this section, we conduct experiments on the application of sparse regression to examine the practical performance of DPONSS. We compare DPONSS with the two state-of-the-art distributed algorithms, i.e., DPOSS [33] and RandGreeDi [1], [27], on both regular-scale and large-scale data sets. The goal of using regular-scale data sets is to examine the approximation ratio and the running time speedup of DPONSS to the centralized PONSS algorithm. Note that regular-scale data sets can be stored on one single machine, and thus can be run by the centralized PONSS algorithm.

To run one distributed algorithm on one data set, the data set is partitioned into m blocks uniformly at random and then distributed to m machines. For each data set, we first generate 10 partitions independently; then, the three distributed algorithms, DPONSS, DPOSS and RandGreeDi, are run on these same partitions, and the average results are reported and compared. Note that when $m=1$, DPONSS, DPOSS and RandGreeDi degenerate to the corresponding centralized algorithms, PONSS, POSS and the greedy algorithm, respectively. Thus, we do not need to perform the partition for the case of $m=1$. But since PONSS and POSS are randomized algorithms, we also repeat running them for 10 times independently and use the average results when $m=1$. Although the greedy algorithm is deterministic, its behavior is randomized under noise, and thus it will also be repeated for 10 times independently.

For DPOSS, we need to run POSS at each machine. The number of iterations for POSS at one machine is set to $2ek^2N$, where N denotes the number of observation variables allocated to that machine, as suggested in [34], [36]. For DPONSS,

PONSS is run at each machine, which has three parameters, B, θ and T. As in [34], we set $B = k$ and $\theta = 0.1$. To make a fair comparison between DPONSS and DPOSS, we use the same number of fitness evaluations for POSS and PONSS at each machine. POSS only needs to evaluate the new offspring solution in each iteration, and thus running $2ek^2N$ iterations implies using $2ek^2N$ evaluations. Note that each iteration of PONSS requires 1 or $(1+2B)$ fitness evaluations, depending on whether the condition "$|Q| = B + 1$" in line 8 of Algorithm 3 is met. Thus, PONSS at each machine keeps running and will be stopped until the $2ek^2N$ number of fitness evaluations is run out.

In the running of each algorithm, to estimate the objective R^2 value of each solution, a sample of instances randomly selected from the whole data set is used. Note that R^2 as presented in Definition 6 captures the portion of the variance of the predictor variable explained by the selected observation variables. The ratio of the sample size over the total number of instances, called *sample ratio*, can be regarded as the noise level. The larger the sample ratio, the lower the noise level. The extreme case is that the sample ratio is 1, which means that the whole data set is used for fitness evaluation, and thus there is no noise. In the experiments, various sample ratios, i.e., various noise levels, will be used. For comparing the final solutions output by each algorithm, the whole data set is used for evaluation. In other words, we can obtain only a noisy objective function value in the optimization process, but for assessing the goodness of the final output solutions, the exact value is used.

To run the experiments, we use an identical configuration: a cluster of 6 machines, where each machine has 16 cores, running Spark 2.2.1. Python 2.7.14 is used for coding.

A. Regular-scale Data Sets

We use 12 regular-scale data sets, as shown in Table I. The two data sets *Cifar10* and *Cifar100* are downloaded from https://www.cs.toronto.edu/~kriz/cifar.html. *VOC2007* and *VOC2012* are downloaded from http://host.robots.ox.ac.uk/pascal/VOC. *Caltech256* and *Caltech101* are downloaded from http://www.vision.caltech.edu/Image_Datasets. The other 6 data sets are downloaded from https://archive.ics.uci.edu/ml/datasets.html and https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Note that some data sets are for classification, including both single-label and multi-label classification, but we use them for regression. For a multi-label classification data set with L labels, the L labels are assigned integers $1, 2, \ldots, L$, respectively. To use it for regression, we construct the target variable of each instance as the sum of the integers corresponding to its labels. Among the 12 data sets in Table I, *VOC2007*, *VOC2012* and *DeliciousMIL* are multi-label classification data sets, each of which has 20 labels. Note that some image data sets, where each instance is an image, are stretched in the experiments. For *VOC2007* and *VOC2012*, the size of each image is not fixed, and we uniformly scale each image such that the size is fixed to be 40×40, i.e., 40 pixels in height and 40 pixels in width. For *Caltech256*, the size of each image is scaled to 48×48. For *Caltech101*, the original size of each image is roughly 300×200, which is scaled to 48×81.

Table I
REGULAR-SCALE DATA SETS FOR SPARSE REGRESSION.

Data set	#inst	#feat
sEMG	1,800	2,500
Cifar10	10,000	3,072
Cifar100	60,000	3,072
SVHN	73,257	3,072
VOC2007	4,952	4,800
VOC2012	11,540	4,800
gisette	7,000	5,000
GGO-Network	2,921	5,232
DS-Activities	9,120	5,625
Caltech256	30,607	6,912
DeliciousMIL	12,234	8,519
Caltech101	9,114	11,664

We set the budget $k = 8$. The number m of reducers is set to $\{1, 2, \ldots, 10\}$. During the running of each algorithm on each data set, when evaluating the fitness of a solution, a random sample of 200 instances is used for estimation. Thus, the sample ratio, i.e., the noise level, is varied for different data sets, ranging from 0.27% on *SVHN* to 11.11% on *sEMG*. We summarize the results in Figure 1, showing the objective R^2 values of the solution output by each algorithm running on each data set for $m \in \{1, 2, \ldots, 10\}$. It can be seen that DPONSS almost always performs the best, except three losses, on *SVHN* with $m = 5$, *VOC2012* with $m = 6$ and *Caltech256* with $m = 1$. This observation thus verifies our theoretical analyses. In [33], it has been shown that DPOSS is better than RANDGREEDI in the noiseless environment. From Figure 1, we can see that the advantage of DPOSS over RANDGREEDI is kept in the noisy environment.

To examine how well DPONSS performs compared to the centralized PONSS algorithm, we calculate the approximation ratio of DPONSS over PONSS. For any $m = i \in \{2, 3, \ldots, 10\}$, the approximation ratio is computed as f_i/f_1, where $\forall i \in \{1, 2, \ldots, 10\}$, f_i denotes the R^2 value of the solution output by DPONSS with $m = i$. Note that f_1 is just the R^2 value output by the centralized PONSS algorithm, because DPONSS with $m = 1$ is just PONSS. The approximation ratios are averaged overall all the 12 data sets, and plotted in Figure 2, showing that the average approximation ratio is at least 97.5% for any m, and can reach 100% for $m = 4, 5$. This implies that the performance of DPONSS is very close to that of the centralized PONSS algorithm. In other words, the distribution leads to little performance decrease. In fact, the distribution can even bring performance increase in some cases, e.g., $m = 6$ on the *Cifar10* data set. This may be because local optima are just avoided by the distribution, i.e., the partition of the whole data set into m blocks.

We also examine the running time speedup of DPONSS over the centralized PONSS algorithm. For any $m = i \in \{2, 3, \ldots, 10\}$, the speedup is computed as t_1/t_i, where $\forall i \in \{1, 2, \ldots, 10\}$, t_i denotes the running time of DPONSS with $m = i$. Note that t_1 is just the running time of the centralized

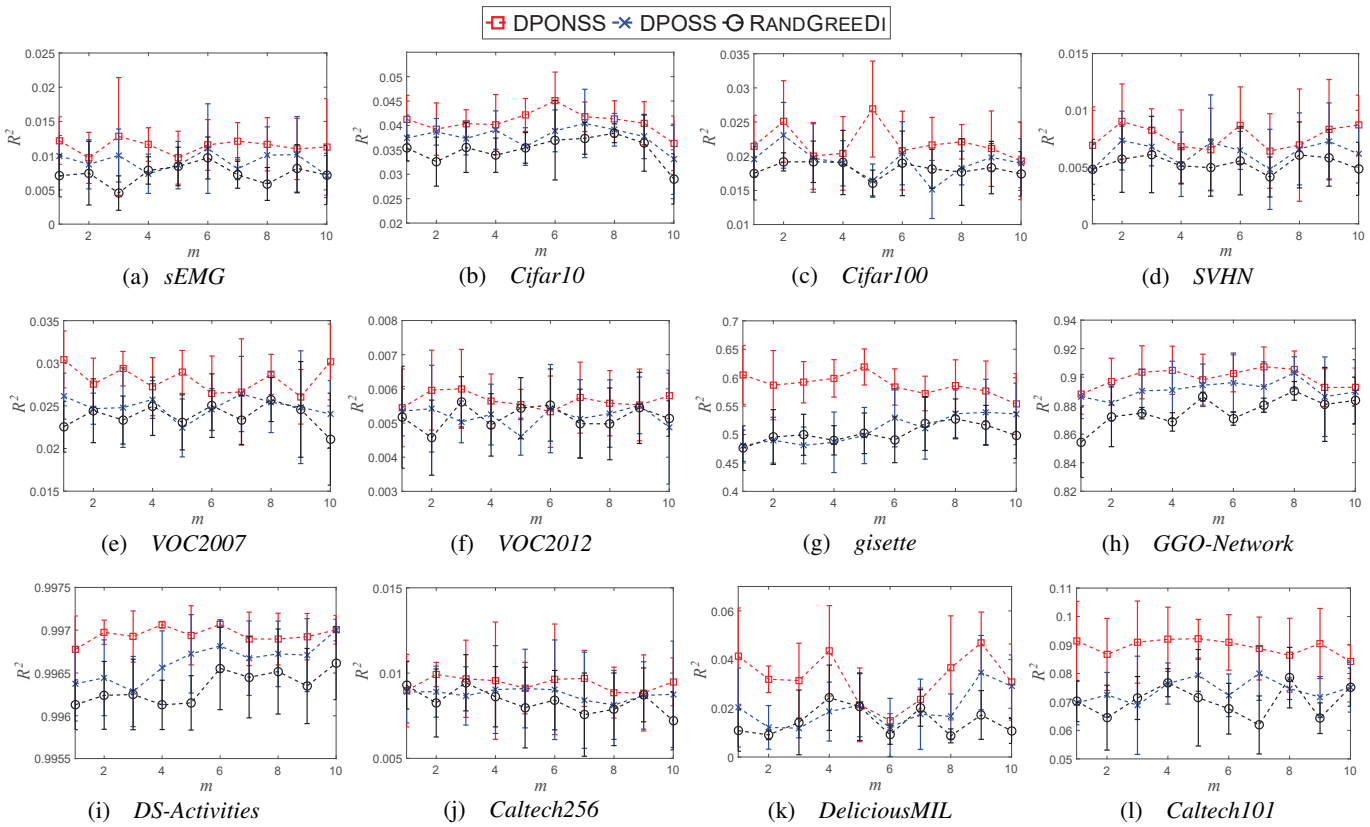

Figure 1. The comparison between DPONSS, DPOSS and RANDGREEDI on 12 regular-scale data sets of sparse regression for $k = 8$ (The objective R^2: the larger, the better).

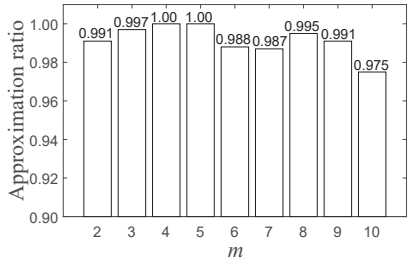

Figure 2. The approximation ratio of DPONSS over the centralized PONSS algorithm at each $m \in \{2, 3, \ldots, 10\}$, which is averaged over all the 12 regular-scale data sets of sparse regression.

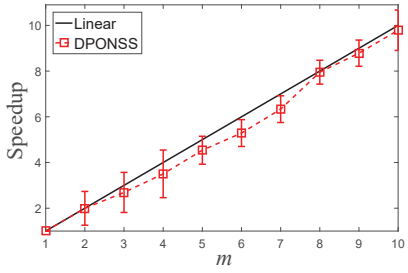

Figure 3. The running time speedup of DPONSS over the centralized PONSS algorithm on the *sEMG* data set.

PONSS algorithm. PONSS uses $2ek^2n$ fitness evaluations in total. For DPONSS, PONSS first uses nearly $2ek^2 \cdot (n/m)$ fitness evaluations at each machine in parallel due to uniform partition, and then uses $2ek^2 \cdot (mk)$ fitness evaluations at one single machine. Thus, when n/m is much larger than mk, the $2ek^2 \cdot (n/m)$ fitness evaluations will dominate the total running time of DPONSS, and linear speedup can be nearly achieved. For the 12 data sets in Table I, it is clear that n/m is much larger than mk for $m \in \{1, 2, \ldots, 10\}$. We plot the running time speedup of DPONSS on the *sEMG* data set in Figure 3, which is consistent with our analysis.

B. Large-scale Data Sets

We use 9 large-scale data sets, as shown in the first three columns of Table II. *UMDFaces* is an image data set, downloaded from https://www.umdfaces.io/. We scale the size of each image to 126×126. *COCO2017* is a multi-label classification data set with 80 labels, downloaded from http://cocodataset.org/#download. *ILSVRC2012* is downloaded from http://www.image-net.org/challenges/LSVRC/2012/. They are two image data sets, and the size of each image is scaled to 256×256. For the three regular-scale image data sets *Caltech101*, *VOC2007* and *VOC2012*, we also stretch them to obtain the corresponding large-scale data sets, named as *Caltech101-large*, *VOC2007-large* and *VOC2012-large*, respectively. The size of each image for these three data sets is uniformly scaled to 266×266, 312×312 and 450×450, respectively. Note that for *Caltech101* with $9,114$ instances, we delete the gray images, and thus, *Caltech101-large* contains only $8,733$ instances. *NYU* is a multi-label classification data set with 894 labels, downloaded from https://cs.nyu.

Table II
THE R^2 VALUE (MEAN±STD.) OF THE FOUR COMPARED ALGORITHMS, DPONSS, DPOSS, RANDGREEDI AND DLS, ON 9 LARGE-SCALE DATA SETS OF SPARSE REGRESSION FOR $k = 8$. ON EACH DATA SET, THE LARGEST VALUES ARE BOLDED. THE COUNT OF DIRECT WIN DENOTES THE NUMBER OF DATA SETS ON WHICH DPONSS HAS A LARGER R^2 VALUE THAN THE CORRESPONDING ALGORITHM (1 TIE IS COUNTED AS 0.5 WIN), WHERE SIGNIFICANT CELLS BY THE SIGN-TEST WITH CONFIDENCE LEVEL 0.05 ARE BOLDED.

Data set	#inst	#feat	DPONSS	DPOSS	RANDGREEDI	DLS
UMDFaces	391,861	47,628	**.0339±.0017**	.0329±.0017	.0314±.0025	.0310±.0020
COCO2017	5,000	196,608	**.0827±.0087**	.0777±.0081	.0705±.0049	.0725±.0107
ILSVRC2012	50,000	196,608	**.0246±.0019**	.0222±.0025	.0223±.0025	.0227±.0030
Caltech101-large	8,733	212,268	**.1237±.0051**	.1204±.0106	.1071±.0094	.1069±.0137
VOC2007-large	4,952	292,032	**.0305±.0025**	.0300±.0036	.0263±.0030	.0253±.0025
VOC2012-large	11,540	607,500	**.0067±.0014**	.0055±.0009	.0057±.0011	.0055±.0015
NYU	1,449	1,017,600	**.0342±.0090**	.0271±.0038	.0258±.0031	.0239±.0048
fgvc-aircraft	3,334	1,062,912	**.0318±.0063**	.0275±.0073	.0241±.0052	.0213±.0067
SUN-od	5,061	1,080,000	**.0507±.0126**	.0467±.0098	.0476±.0131	.0395±.0065
DPONSS: Count of direct win			-	**9**	**9**	**9**
Average rank			1	2.5	2.9	3.6

edu/~silberman/datasets/nyu_depth_v2.html. The size of each image is scaled to 640×530. *fgvc-aircraft* is an image data set downloaded from http://www.robots.ox.ac.uk/~vgg/data/fgvc-aircraft/#format, where each image is scaled from size 692×1024 to 692×512. Note that there are four variables, 'model', 'variant', 'family' and 'manufacturer', to predict, and we only use 'variant'. *SUN-od* is a multi-label image data set with $2,776$ labels, downloaded from http://groups.csail.mit.edu/vision/SUN/. The size of each image is scaled to 600×600. To use multi-label large-scale data sets for regression, the target variable is constructed in the same way as that for multi-label regular-scale data sets.

For the comparison on large-scale data sets, we also implement the distributed version of LS [30], called DLS, by applying the divide and conquer idea. The budget k is also set to 8. We use different number of reducers for large-scale data sets. The larger the data set, the more the number m of reducers. For the first five data sets in Table II, m is set to 300; for *VOC2012-large*, $m = 600$; for the last three data sets, *NYU*, *fgvc-aircraft* and *SUN-od*, $m = 1,000$. As the number m of reducers exceeds the total number 6×16 of cores, each core will carry out several reduce tasks sequentially. To estimate the fitness of a solution during the running of each algorithm, the sample ratios employed for the 9 large-scale data sets in Table II are 1%, 10%, 3%, 10%, 10%, 5%, 20%, 5% and 3%, respectively.

The results are summarized in Table II, showing that DPONSS always achieves the best performance. By the *sign-test* [8] with confidence level 0.05, DPONSS is significantly better than the other three algorithms. The rank of each algorithm on each data set is also computed as in [8], and averaged in the last row of Table II.

The sample ratio, i.e., the ratio of the sample size over the total number of instances, characterizes the noise level. We examine the influence of the sample ratio on the performance of DPONSS. Although the sample ratios for different data sets used in our experiments can be different, it requires to test the performance of DPONSS on the same data set with different sample ratios. On the *COCO2017* data set, we vary the sample ratio from 8% to 16%. The results in Figure 4 show that DPONSS is always the best, implying that DPONSS can achieve good performance under different noise levels.

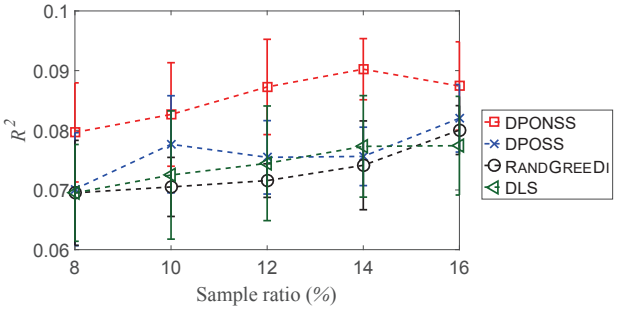

Figure 4. The comparison between DPONSS, DPOSS, RANDGREEDI and DLS on the *COCO2017* data set for the sample ratio in $\{8\%, 10\%, \ldots, 16\%\}$.

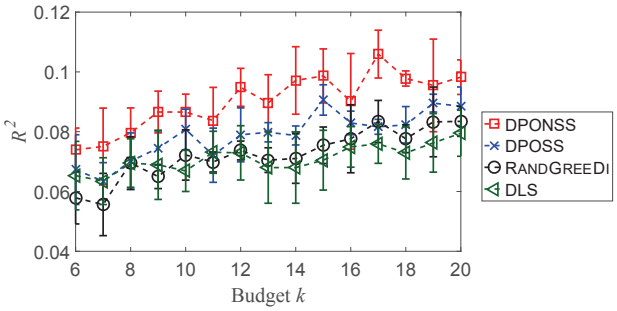

Figure 5. The comparison between DPONSS, DPOSS, RANDGREEDI and DLS on the *COCO2017* data set for the budget $k \in \{6, 7, \ldots, 20\}$.

We then also examine the influence of the budget k. On *COCO2017* with the sample ratio 8%, we vary k from 6 to 20. The results in Figure 5 show that DPONSS keeps the best.

VI. CONCLUSION

This paper studies the problem of large-scale noisy subset selection. First, we analyze the performance of DPOSS, a state-of-the-art distributed algorithm based on multi-objective evolutionary optimization for large-scale subset selection, under noise. The derived approximation guarantees imply that the noise leads to a large performance decrease. Then, we propose the distributed algorithm DPONSS, which employs a noise-aware multi-objective evolutionary algorithm PONSS at each machine. We prove that DPONSS achieves significantly better approximation guarantees than DPOSS under noise. For the application of sparse regression, the performance of DPONSS

is empirically tested using Spark on real-world data sets, where the size ranges from thousands to millions. The results show the excellent performance of DPONSS. In the future, we will apply DPONSS to the data sets with size tens of millions, and also to more large-scale noisy applications of subset selection.

ACKNOWLEDGMENTS

This research was supported by the National Key Research and Development Program of China (2017YFB1003102) and the National Natural Science Foundation of China (61603367).

REFERENCES

[1] R. Barbosa, A. Ene, H. Nguyen, and J. Ward, "The power of randomization: Distributed submodular maximization on massive datasets," in *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, Lille, France, 2015, pp. 1236–1244.

[2] T. Bartz-Beielstein and S. Markon, "Threshold selection, hypothesis tests, and DOE methods," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC'02)*, Honolulu, HI, 2002, pp. 777–782.

[3] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, "Guarantees for greedy maximization of non-submodular functions with applications," in *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, Sydney, Australia, 2017, pp. 498–507.

[4] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

[5] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'10)*, Washington, DC, 2010, pp. 1029–1038.

[6] 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)*, Bellevue, WA, 2011, pp. 1057–1064.

[7] ——, "Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection," *Journal of Machine Learning Research*, vol. 19, pp. 1–34, 2018.

[8] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

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

[10] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen, "Evolution of repertoire-based control for robots with complex locomotor systems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 314–328, 2018.

[11] D. Dueck and B. J. Frey, "Non-metric affinity propagation for unsupervised image categorization," in *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV'07)*, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[12] E. R. Elenberg, R. Khanna, A. G. Dimakis, S. Negahban *et al.*, "Restricted strong convexity implies weak submodularity," *The Annals of Statistics*, vol. 46, no. 6B, pp. 3539–3568, 2018.

[13] A. K. Farahat, A. Ghodsi, and M. S. Kamel, "An efficient greedy method for unsupervised feature selection," in *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11)*, Vancouver, Canada, 2011, pp. 161–170.

[14] U. Feige, "A threshold of $\ln n$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[15] C. Feng, C. Qian, and K. Tang, "Unsupervised feature selection by Pareto optimization," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, Honolulu, HI, 2019.

[16] T. Friedrich, T. Kötzing, M. S. Krejca, and A. M. Sutton, "The compact genetic algorithm is efficient under extreme gaussian noise," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 477–490, 2017.

[17] T. Friedrich and F. Neumann, "Maximizing submodular functions under matroid constraints by evolutionary algorithms," *Evolutionary Computation*, vol. 23, no. 4, pp. 543–558, 2015.

[18] A. Hassidim and Y. Singer, "Submodular optimization under noise," in *Proceedings of the 30th Conference on Learning Theory (COLT'17)*, Amsterdam, The Netherlands, 2017, pp. 1069–1122.

[19] T. Horel and Y. Singer, "Maximization of approximately submodular functions," in *Advances in Neural Information Processing Systems 29 (NIPS'16)*, Barcelona, Spain, 2016, pp. 3045–3053.

[20] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-A survey," *IEEE Transactions on evolutionary computation*, vol. 9, no. 3, pp. 303–317, 2005.

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

[22] 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)*, Washington, DC, 2003, pp. 137–146.

[23] 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*, vol. 9, pp. 235–284, 2008.

[24] X. Lai, Y. Zhou, J. He, and J. Zhang, "Performance analysis of evolutionary algorithms for the minimum label spanning tree problem," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 860–872, 2014.

[25] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 170–182, 2004.

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

[27] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, "Distributed submodular maximization: Identifying representative elements in massive data," in *Advances in Neural Information Processing Systems 26 (NIPS'13)*, Lake Tahoe, NV, 2013, pp. 2049–2057.

[28] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995.

[29] G. L. Nemhauser and L. A. Wolsey, "Best algorithms for approximating the maximum of a submodular set function," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 177–188, 1978.

[30] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions – I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[31] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward, "Genetic improvement of software: A comprehensive survey," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 415–432, 2018.

[32] C. Pizzuti, "Evolutionary computation for community detection in networks: A review," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 464–483, 2018.

[33] C. Qian, G. Li, C. Feng, and K. Tang, "Distributed Pareto optimization for subset selection," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, Stockholm, Sweden, 2018, pp. 1492–1498.

[34] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou, "Subset selection under noise," in *Advances in Neural Information Processing Systems 30 (NIPS'17)*, Long Beach, CA, 2017, pp. 3562–3572.

[35] C. Qian, Y. Yu, and K. Tang, "Approximation guarantees of stochastic greedy algorithms for subset selection," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, Stockholm, Sweden, 2018, pp. 1478–1484.

[36] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by Pareto optimization," in *Advances in Neural Information Processing Systems 28 (NIPS'15)*, Montreal, Canada, 2015, pp. 1765–1773.

[37] C. Qian, J.-C. Shi, K. Tang, and Z.-H. Zhou, "Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 595–608, 2018.

[38] C. Qian, Y. Yu, and Z.-H. Zhou, "Analyzing evolutionary optimization in noisy environments," *Evolutionary computation*, vol. 26, no. 1, pp. 1–41, 2018.

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

[40] Á. Rubio-Largo, L. Vanneschi, M. Castelli, and M. A. Vega-Rodríguez, "Multiobjective metaheuristic to design RNA sequences," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 156–169, 2019.

[41] A. Singla, S. Tschiatschek, and A. Krause, "Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, Phoenix, AZ, 2016, pp. 2037–2043.

[42] M.-H. Tayarani-N, X. Yao, and H. Xu, "Meta-heuristic algorithms in car engine design: A literature survey," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 609–629, 2015.

[43] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, Phoenix, AZ, 2016, pp. 819–826.

Chao Qian received the BSc and PhD degrees in computer science from Nanjing University, China, in 2009 and 2015, respectively. Since then, he joined the School of Computer Science & Technology at University of Science and Technology of China as an associate researcher. His research interests are mainly in evolutionary computation and machine learning, particularly, the theoretical foundation of evolutionary algorithms and its application with theoretical guarantees in machine learning. He has published over 20 first-author papers in leading international journals and conference proceedings, including *Artificial Intelligence*, *Evolutionary Computation*, *IEEE Transactions on Evolutionary Computation*, *Algorithmica*, NIPS, IJCAI, AAAI, etc. He won the ACM GECCO 2011 Best Paper Award (Theory Track), the IDEAL 2016 Best Paper Award, and the 2017 Outstanding Doctoral Dissertation Award of CAAI. He has served as chair of the IEEE Computational Intelligence Society Task Force "Theoretical Foundations of Bio-inspired Computation", and an Young Associate Editor of *Frontiers of Computer Science*. He has also been selected to the Young Elite Scientists Sponsorship Program by CAST.