Subset Selection by Pareto Optimization with Recombination

Chao Qian,[1]* **Chao Bian,**[1] **Chao Feng**[2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
qianc@lamda.nju.edu.cn, chaobian12@gmail.com, chaofeng@mail.ustc.edu.cn

Abstract

Subset selection, i.e., to select a limited number of items optimizing some given objective function, is a fundamental problem with various applications such as unsupervised feature selection and sparse regression. By employing a multi-objective evolutionary algorithm (EA) with mutation only to optimize the given objective function and minimize the number of selected items simultaneously, the recently proposed POSS algorithm achieves state-of-the-art performance for subset selection. In this paper, we propose the PORSS algorithm by incorporating recombination, a characterizing feature of EAs, into POSS. We prove that PORSS can achieve the optimal polynomial-time approximation guarantee as POSS when the objective function is monotone, and can find an optimal solution efficiently in some cases whereas POSS cannot. Extensive experiments on unsupervised feature selection and sparse regression show the superiority of PORSS over POSS. Our analysis also theoretically discloses that recombination from diverse solutions can be more likely than mutation alone to generate various variations, thereby leading to better exploration; this may be of independent interest for understanding the influence of recombination.

Introduction

This paper considers a general problem, i.e., subset selection, which is to select a subset of size at most k from a total set of n items for maximizing (or minimizing) some given objective function f. This problem arises in various real-world applications, such as maximum coverage (Feige 1998), sparse regression (Miller 2002), influence maximization (Kempe, Kleinberg, and Tardos 2003), sensor placement (Krause, Singh, and Guestrin 2008), document summarization (Lin and Bilmes 2011) and unsupervised feature selection (Farahat, Ghodsi, and Kamel 2011), to name a few.

Subset selection is generally NP-hard, and much efforts have been devoted to developing polynomial-time approximation algorithms. The greedy algorithm, which iteratively selects one item with the largest marginal gain, has been shown to be a good approximation solver. When the involved objective function f satisfies the monotone property, the greedy algorithm can achieve the $(1-e^{-\gamma})$-approximation guarantee, where γ is the submodularity ratio measuring how close f is to submodularity (Das and Kempe 2011). Particularly, for submodular objective functions, $\gamma = 1$ and the approximation guarantee becomes $1 - 1/e$, which is optimal, i.e., cannot be improved by any polynomial-time algorithm (Nemhauser and Wolsey 1978). Harshaw *et al.* (2019) have recently proved that the general approximation guarantee of $(1 - e^{-\gamma})$ is also optimal.

Based on Pareto optimization, Qian *et al.* (2015) proposed the POSS algorithm for subset selection. The idea is to reformulate subset selection as a bi-objective optimization problem maximizing the given objective and minimizing the subset size simultaneously, then solve the problem by a multi-objective EA (MOEA), and finally select the best solution with size at most k from the generated solution set. It has been shown that POSS can achieve the optimal polynomial-time approximation guarantee, $1-e^{-\gamma}$, and can be significantly better than the greedy algorithm in applications, e.g., unsupervised feature selection and sparse regression. Moreover, POSS is robust against uncertainties (Qian et al. 2017; Roostapour et al. 2019), and easily distributed for large-scale tasks (Qian et al. 2016; 2018; Qian 2019).

The optimization engine of POSS is the employed MOEA, which iteratively reproduces new solutions for solving the reformulated bi-objective problem. For EAs, *mutation* and *recombination* (or called *crossover*) are two popular operators for reproduction (Bäck 1996); the former changes one solution randomly whereas the latter mixes up two or more solutions. POSS applies mutation only and has performed well, while recombination, as a core feature of EAs, may be helpful to further improve its performance.

In this paper, we propose the PORSS algorithm for subset selection by introducing recombination into POSS. Two common recombination operators are considered: one-point recombination and uniform recombination. In theory, we prove that for subset selection with monotone objective functions, PORSS can achieve the optimal polynomial-time approximation guarantee, $1-e^{-\gamma}$; for one concrete example of subset selection, PORSS can be significantly faster than POSS to find an optimal solution. We also conduct experiments on the applications of unsupervised feature selection and sparse regression with various real-world data sets, showing that within the same running time, PORSS can almost always achieve better performance than POSS.

*This work was supported by the NSFC (61603367).

Note that recombination is understood only at preliminary level, though there are great efforts devoted to analyzing its influence, e.g., (Neumann and Theile 2010; Doerr et al. 2013; Qian, Yu, and Zhou 2013; Oliveto and Witt 2014; Sudholt 2017; Dang et al. 2018). Our analysis theoretically discloses that recombining diverse solutions is more likely than mutation to generate various variations, and thus to escape from local optima; this may help to understand this kind of operator.

Subset Selection

Given a ground set $V = \{v_1, v_2, \dots, v_n\}$, we study the functions $f : 2^V \to \mathbb{R}$ over subsets of V. A set function f is monotone if $\forall S \subseteq T$, $f(S) \leq f(T)$. Assume w.l.o.g. that monotone functions are normalized, i.e., $f(\emptyset) = 0$. A set function f is submodular (Nemhauser, Wolsey, and Fisher 1978) if $\forall S \subseteq T \subseteq V$,

$$f(T) - f(S) \leq \sum\nolimits_{v \in T \setminus S} \big(f(S \cup \{v\}) - f(S)\big).$$

For a general set function f, the notion of submodularity ratio in Definition 1 is used to measure to what extent f has the submodular property. When f is monotone, it holds that (1) $\forall S, l : 0 \leq \gamma_{S,l}(f) \leq 1$, and (2) f is submodular iff $\forall S, l : \gamma_{S,l}(f) = 1$.

Definition 1 (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*

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

The subset selection problem as presented in Definition 2 is to select a subset S of V such that a given objective f is maximized with the constraint $|S| \leq k$. For a monotone 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 $(1 - e^{-\gamma_{S,k}(f)})$ (Das and Kempe 2011), where S is the subset output by the greedy algorithm. The optimality of this approximation guarantee was known only in the case where $\gamma_{S,k}(f) = 1$, i.e., f is submodular (Nemhauser and Wolsey 1978), and has recently been proved in the general case (Harshaw et al. 2019).

Definition 2 (Subset Selection)**.** *Given all items $V = \{v_1, v_2, \dots, v_n\}$, an objective function f and a budget k, to find a subset of at most k items maximizing f, i.e.,*

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

Here are two applications of subset selection with monotone, but not necessarily submodular, objective functions, that will be studied in this paper. Unsupervised feature selection as presented in Definition 3 is to select at most k columns from a matrix $\mathbf{A}$ to best approximate $\mathbf{A}$. Some notations: $(\cdot)^+$: Moore-Penrose inverse of a matrix; $\|\cdot\|_F$: Frobenius norm of a matrix; $|\cdot|$: number of columns of a matrix. The goodness of approximation is measured by the sum of squared errors between the original matrix $\mathbf{A}$ and the approximation $\mathbf{SS}^+\mathbf{A}$, where $\mathbf{SS}^+$ is the projection matrix onto the space spanned by the columns of $\mathbf{S}$. Note that a submatrix of $\mathbf{A}$ can be seen as a subset of all columns of $\mathbf{A}$.

Definition 3 (Unsupervised Feature Selection)**.** *Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a budget k, to find a submatrix $\mathbf{S}$ of $\mathbf{A}$ with at most k columns minimizing $\|\mathbf{A} - \mathbf{SS}^+\mathbf{A}\|_F^2$, i.e.,*

$$\arg\min\nolimits_{\mathbf{S}\textit{: a submatrix of } \mathbf{A}} \|\mathbf{A} - \mathbf{SS}^+\mathbf{A}\|_F^2 \quad \textit{s.t.} \quad |\mathbf{S}| \leq k.$$

For the ease of theoretical treatment, this minimization problem is often equivalently reformulated as a maximization problem (Bhaskara et al. 2016; Ordozgoiti, Canaval, and Mozo 2018):

$$\arg\max\nolimits_{\mathbf{S}\textit{: a submatrix of } \mathbf{A}} \|\mathbf{SS}^+\mathbf{A}\|_F^2 \quad \textit{s.t.} \quad |\mathbf{S}| \leq k.$$

Sparse regression (Miller 2002) as presented in Definition 4 is to find a sparse approximation solution to the linear regression problem. Note that S and its index set $\{i \mid v_i \in S\}$ are not distinguished for notational convenience, and all variables are assumed w.l.o.g. to be normalized to have expectation 0 and variance 1.

Definition 4 (Sparse Regression)**.** *Given observation variables $V = \{v_1, \dots, v_n\}$, a predictor variable z and a budget k, to find at most k variables from V maximizing the* squared multiple correlation *(Johnson and Wichern 2007), i.e.,*

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

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

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

The POSS Algorithm

Based on Pareto optimization, a new algorithm POSS for subset selection has been proposed (Friedrich and Neumann 2015; Qian, Yu, and Zhou 2015). Note that a subset S of V can be represented by a binary vector $\boldsymbol{x} \in \{0,1\}^n$, where $x_i = 1$ iff the item $v_i \in S$, and we will not distinguish them for notational convenience. POSS reformulates the original problem Eq. (1) as a bi-objective minimization problem:

$$\arg\min\nolimits_{\boldsymbol{x} \in \{0,1\}^n} \quad (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})), \tag{2}$$

where

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

Thus, POSS maximizes the original objective function f and minimizes the subset size $|\boldsymbol{x}|$ simultaneously. By setting f_1 to $+\infty$ for $|\boldsymbol{x}| \geq 2k$, overly *infeasible* solutions, i.e., solutions with large constraint violation, are excluded.

To compare solutions in bi-objective optimization, POSS uses the domination relationship. For two solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\boldsymbol{x}$ *weakly dominates* $\boldsymbol{x}'$, denoted as $\boldsymbol{x} \preceq \boldsymbol{x}'$, if $f_1(\boldsymbol{x}) \leq f_1(\boldsymbol{x}') \wedge f_2(\boldsymbol{x}) \leq f_2(\boldsymbol{x}')$; $\boldsymbol{x}$ *dominates* $\boldsymbol{x}'$, denoted as $\boldsymbol{x} \prec \boldsymbol{x}'$, if $\boldsymbol{x} \preceq \boldsymbol{x}'$ and either $f_1(\boldsymbol{x}) < f_1(\boldsymbol{x}')$ or $f_2(\boldsymbol{x}) < f_2(\boldsymbol{x}')$; they are *incomparable*, if neither $\boldsymbol{x} \preceq \boldsymbol{x}'$ nor $\boldsymbol{x}' \preceq \boldsymbol{x}$.

POSS employs a simple MOEA with mutation only, which is slightly modified from the GSEMO algorithm (Giel

Algorithm 1 POSS Algorithm

```
Input: $V = \{v_1, \ldots, v_n\}$; objective $f : 2^V \to \mathbb{R}$; budget $k$
Parameter: the number $T$ of iterations
Output: a subset of $V$ with at most $k$ items
Process:
 1: Let $\boldsymbol{x} = 0^n$, $P = \{\boldsymbol{x}\}$ and $t = 0$;
 2: while $t < T$ do
 3:     Select $\boldsymbol{x}$ from $P$ randomly;
 4:     Apply bit-wise mutation on $\boldsymbol{x}$ to generate $\boldsymbol{x}'$;
 5:     if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec \boldsymbol{x}'$ then
 6:         $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{x}' \preceq \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$
 7:     end if
 8:     $t = t + 1$
 9: end while
10: return $\arg\max_{\boldsymbol{x} \in P, |\boldsymbol{x}| \le k} f(\boldsymbol{x})$
```

2003; Laumanns, Thiele, and Zitzler 2004), to solve the bi-objective problem Eq. (2). As described in Algorithm 1, it starts from 0^n representing the empty set, and iteratively tries to improve the solutions in the population P (lines 2-9). In each iteration, a solution $\boldsymbol{x}$ is selected from P uniformly at random, and used to generate a new solution $\boldsymbol{x}'$ by the bit-wise mutation operator, presented as follows:

Bit-wise mutation: flip each bit of a solution $\boldsymbol{x} \in \{0, 1\}^n$ independently with probability $1/n$.

The newly generated solution $\boldsymbol{x}'$ is then used to update P in lines 5-7, making P contain only non-dominated solutions generated-so-far. That is, if $\boldsymbol{x}'$ is not dominated by any solution in P (line 5), it will be added into P, and meanwhile those archived solutions weakly dominated by $\boldsymbol{x}'$ will be deleted (line 6). After running T iterations, the best solution w.r.t. the original problem Eq. (1) is selected from P in line 10 as the final output solution.

For subset selection with monotone objective functions, POSS has been proved to achieve the same general approximation guarantee as the greedy algorithm in polynomial expected running time, i.e., to achieve the optimal polynomial-time approximation guarantee (Qian, Yu, and Zhou 2015). Furthermore, it has been empirically shown that POSS can achieve significantly better performance than the greedy algorithm in some applications, e.g., unsupervised feature selection (Feng, Qian, and Tang 2019) and sparse regression (Qian, Yu, and Zhou 2015).

The PORSS Algorithm

To reproduce new solutions in each iteration, POSS applies the mutation operator, which simulates the mutation phenomena in DNA transformation. It is known that recombination is another popular operator for reproduction, which simulates the chromosome exchange phenomena in zoogamy reproduction, and typically appears in various real EAs, e.g., the popularly used algorithm NSGA-II (Deb et al. 2002).

In this section, we propose a new Pareto Optimization algorithm with Recombination for Subset Selection, briefly called PORSS. As described in Algorithm 2, PORSS employs recombination and mutation together, rather than mutation only, to generate new solutions in each iteration. In line 3, two solutions are selected randomly from the population P with replacement, and then recombined to generate new solutions in line 4. We consider two commonly used recombination operators:

One-point recombination: select $i \in \{1, 2, \ldots, n\}$ randomly, and exchange the first i bits of two solutions;

Uniform recombination: exchange each bit of two solutions independently with probability $1/2$.

For example, for solutions 0^n and 1^n, two new solutions $0^{n/2}1^{n/2}$ and $1^{n/2}0^{n/2}$ can be generated by one-point recombination with probability $1/n$, and by uniform recombination with probability $(1/2^n) \cdot 2$, where the factor 2 is included due to the symmetry. In line 5, the two solutions generated by recombination are further mutated to generate another two ones, which are used to update the population P in lines 6-10.

Algorithm 2 PORSS Algorithm

```
Input: $V = \{v_1, \ldots, v_n\}$; objective $f : 2^V \to \mathbb{R}$; budget $k$
Parameter: the number $T$ of iterations
Output: a subset of $V$ with at most $k$ items
Process:
 1: Let $\boldsymbol{x} = 0^n$, $P = \{\boldsymbol{x}\}$ and $t = 0$;
 2: while $t < T$ do
 3:     Select $\boldsymbol{x}, \boldsymbol{y}$ from $P$ randomly with replacement;
 4:     Apply recombination on $\boldsymbol{x}, \boldsymbol{y}$ to generate $\boldsymbol{x}'$, $\boldsymbol{y}'$;
 5:     Apply bit-wise mutation on $\boldsymbol{x}', \boldsymbol{y}'$ to generate $\boldsymbol{x}'', \boldsymbol{y}''$;
 6:     for each $\boldsymbol{q} \in \{\boldsymbol{x}'', \boldsymbol{y}''\}$
 7:         if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec \boldsymbol{q}$ then
 8:             $P = (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{q} \preceq \boldsymbol{z}\}) \cup \{\boldsymbol{q}\}$
 9:         end if
10:     end for
11:     $t = t + 1$
12: end while
13: return $\arg\max_{\boldsymbol{x} \in P, |\boldsymbol{x}| \le k} f(\boldsymbol{x})$
```

Influence of Recombination

To understand the influence of recombination intuitively, we compare the distribution of the number of bits flipped with/without recombination. Suppose two solutions $\boldsymbol{x}, \boldsymbol{y}$ selected in line 3 of Algorithm 2 have Hamming distance d, denoted as $H(\boldsymbol{x}, \boldsymbol{y}) = d$. Let $\boldsymbol{x}'', \boldsymbol{y}''$ denote the two solutions generated by recombination and mutation in line 5. We analyze the probability for $\boldsymbol{x}''$ or $\boldsymbol{y}''$ to have Hamming distance j with $\boldsymbol{x}$, denoted as $Q(d, j)$. That is,

$$Q(d, j) = \mathrm{P}(H(\boldsymbol{x}, \boldsymbol{x}'') = j \vee H(\boldsymbol{x}, \boldsymbol{y}'') = j \mid H(\boldsymbol{x}, \boldsymbol{y}) = d).$$

For one-point and uniform recombination, we use the notations $Q_o(d, j)$ and $Q_u(d, j)$, respectively. Let $\boldsymbol{z}_m$ denote the solution generated from a solution $\boldsymbol{z}$ by bit-wise mutation. By turning off recombination, i.e., deleting line 4 of Algorithm 2, we analyze the corresponding probability

$$Q_m(d, j) = \mathrm{P}(H(\boldsymbol{x}, \boldsymbol{x}_m) = j \vee H(\boldsymbol{x}, \boldsymbol{y}_m) = j | H(\boldsymbol{x}, \boldsymbol{y}) = d).$$

In the following, we compare $Q_o(d, j)$, $Q_u(d, j)$ with $Q_m(d, j)$, to examine the influence of recombination.

Given a solution $\boldsymbol{z}$ with Hamming distance i from $\boldsymbol{x}$, let $q_{i,j}$ denote the probability for the Hamming distance to become j by bit-wise mutation, i.e.,

$$q_{i,j} = \mathrm{P}(H(\boldsymbol{x}, \boldsymbol{z}_m) = j \mid H(\boldsymbol{x}, \boldsymbol{z}) = i).$$

For $Q_m(d,j)$, as $H(\boldsymbol{x}, \boldsymbol{x}) = 0$ and $H(\boldsymbol{x}, \boldsymbol{y}) = d$, we have

$$Q_m(d,j) = q_{0,j} + q_{d,j} - q_{0,j} \cdot q_{d,j}.$$

By uniform recombination, $\boldsymbol{x}, \boldsymbol{y}$ exchange i different bits with probability $\binom{d}{i}(1/2)^i(1/2)^{d-i} = \binom{d}{i}(1/2)^d$, generating two solutions $\boldsymbol{x}', \boldsymbol{y}'$ where $H(\boldsymbol{x}, \boldsymbol{x}') = i$ and $H(\boldsymbol{x}, \boldsymbol{y}') = d - i$. Note that $\boldsymbol{x}$ and $\boldsymbol{y}$ have totally d different bits. Considering the mutation behavior on $\boldsymbol{x}', \boldsymbol{y}'$, we have

$$Q_u(d,j) = \sum_{i=0}^{d} \binom{d}{i} \frac{1}{2^d} \cdot (q_{i,j} + q_{d-i,j} - q_{i,j} \cdot q_{d-i,j}).$$

Consider one-point recombination. $\forall 1 \le i \le d$, there exists $l \ge i$ such that exchanging the first l bits of $\boldsymbol{x}, \boldsymbol{y}$ can generate two solutions $\boldsymbol{x}', \boldsymbol{y}'$ where $H(\boldsymbol{x}, \boldsymbol{x}') = i$ and $H(\boldsymbol{x}, \boldsymbol{y}') = d - i$. Thus, we have

$$Q_o(d,j) \ge \frac{1}{n} \sum_{i=1}^{d} (q_{i,j} + q_{d-i,j} - q_{i,j} \cdot q_{d-i,j}).$$

Because it is sufficient to keep all bits unchanged in mutation, $q_{j,j} \ge (1 - 1/n)^n \ge 1/(2e)$. Thus, we have

$(a)\ \forall j \le d : Q_o(d,j) \ge 1/n \cdot q(j,j) = \Omega(1/n);$

$(b)\ \forall j \le d : Q_u(d,j) \ge \binom{d}{j} \frac{1}{2^d} \cdot q(j,j) = \Omega\left(\binom{d}{j} / 2^d\right).$

By analyzing $q_{0,j}$ and $q_{d,j}$, we can derive that, $\exists 0 < j_0 \le d$,

$(c.1)$ for $j < j_0 : \Omega\left((1/j)^j\right) \le Q_m(d,j) \le O\left((e/j)^j\right);$

$(c.2)$ for $j \ge j_0 :$

$$\Omega\left(\frac{1}{d-j} \cdot \frac{d}{n}\right)^{d-j} \le Q_m(d,j) \le O\left(\frac{e}{d-j} \cdot \frac{d}{n}\right)^{d-j}.$$

The detailed analysis for $Q_m(d,j)$ is provided in the supplementary material due to space limitations.

According to (c.1) and (c.2), the number of bits flipped by mutation only is strongly concentrated around two extreme values, 0 and d. When j increases from 0 to j_0 or decreases from d to j_0, $Q_m(d,j)$ decays super-exponentially. Particularly, for $j = d/2$, $q(0, d/2) \le \binom{n}{d/2}(1/n)^{d/2} \le 1/(d/2)!$, $q(d, d/2) \le \binom{d}{d/2}(1/n)^{d/2} \le 1/(d/2)!$, and thus,

$$Q_m(d, d/2) \le \frac{2}{(d/2)!} \le \frac{2}{e(d/(2e))^{d/2}} \le \left(\frac{2e}{d}\right)^{d/2}, \quad (3)$$

where the second inequality holds by Stirling's formula. According to (b), the number of bits flipped by uniform recombination and mutation is concentrated around $d/2$, but $Q_u(d,j)$ is always lower bounded by $\Omega(1/2^d)$, which is much greater than $Q_m(d, d/2)$ in Eq. (3) when d is large. According to (a), $\forall j \le d : Q_o(d,j) \ge \Omega(1/n)$, implying that the number of bits flipped by one-point recombination and mutation is relatively uniformly distributed.

Therefore, from diverse solutions, i.e., when d is large, recombination can ease flipping any number of bits, and may lead to better exploration and thus a better ability of escaping from local optima. The advantage of recombination will be verified by theoretical analysis and empirical study.

Theoretical Analysis

As introduced before, the greedy algorithm and POSS can achieve the optimal polynomial-time approximation guarantee for subset selection with monotone objective functions. A natural question is whether PORSS can keep the optimal approximation. We give the positive answer by proving Theorem 1, i.e., PORSS achieves the approximation guarantee of $(1 - e^{-\gamma_{\min}})$ in polynomial expected running time. Let OPT denote the optimal function value. The proof is inspired by the analysis of POSS (Qian, Yu, and Zhou 2015).

Lemma 1. *(Qian et al. 2016) Let $f : \{0,1\}^n \to \mathbb{R}^+$ be a monotone function. For any $\boldsymbol{x} \in \{0,1\}^n$, there exists one item $v \notin \boldsymbol{x}$ such that*

$$f(\boldsymbol{x} \cup \{v\}) - f(\boldsymbol{x}) \ge \frac{\gamma_{\boldsymbol{x},k}}{k}(\mathrm{OPT} - f(\boldsymbol{x})).$$

Theorem 1. *For subset selection with any monotone f, the expected number of iterations until PORSS with one-point or uniform recombination finds a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \le k$ and $f(\boldsymbol{x}) \ge (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$ is polynomial, where $\gamma_{\min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$.*

Proof. Let $J_{\max}$ be the maximum value of $j \in \{0, 1, \ldots, k\}$ such that in the population P, there exists a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \le j$ and $f(\boldsymbol{x}) \ge (1 - (1 - \gamma_{\min}/k)^j) \cdot \mathrm{OPT}$. That is,

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

We only need to analyze the expected number of iterations until $J_{\max} = k$, which implies that there exists one solution $\boldsymbol{x} \in P$ satisfying that $|\boldsymbol{x}| \le k$ and $f(\boldsymbol{x}) \ge (1 - (1 - \gamma_{\min}/k)^k) \cdot \mathrm{OPT} \ge (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$.

As PORSS starts from 0^n, $J_{\max}$ is initially 0. Assume that currently $J_{\max} = i < k$. Let $\boldsymbol{x}$ denote a solution corresponding to $J_{\max} = i$, i.e., $|\boldsymbol{x}| \le i$ and $f(\boldsymbol{x}) \ge (1 - (1 - \gamma_{\min}/k)^i) \cdot \mathrm{OPT}$. First, $J_{\max}$ will not decrease. This is because deleting $\boldsymbol{x}$ from P in line 8 of Algorithm 2 implies that $\boldsymbol{x}$ is weakly dominated by the newly included solution $\boldsymbol{q}$, satisfying that $|\boldsymbol{q}| \le |\boldsymbol{x}| \le i$ and $f(\boldsymbol{q}) \ge f(\boldsymbol{x}) \ge (1 - (1 - \gamma_{\min}/k)^i) \cdot \mathrm{OPT}$.

Next, we analyze the probability of increasing $J_{\max}$ in one iteration. Consider the case that the two selected solutions in line 3 of Algorithm 2 are both $\boldsymbol{x}$, occurring with probability $(1/|P|) \cdot (1/|P|)$ due to uniform selection with replacement. For two identical solutions, either one-point or uniform recombination in line 4 makes no changes. Thus, in line 5, $\boldsymbol{x}$ is used to generate a new solution by bit-wise mutation, and this process is implemented twice independently. For bit-wise mutation on $\boldsymbol{x}$, according to Lemma 1, a new solution $\boldsymbol{x}'$ satisfying $f(\boldsymbol{x}') - f(\boldsymbol{x}) \ge (\gamma_{\boldsymbol{x},k}/k) \cdot (\mathrm{OPT} - f(\boldsymbol{x}))$ can be generated by flipping only one specific 0 bit of $\boldsymbol{x}$ (i.e., adding a specific item into $\boldsymbol{x}$), occurring with probability $(1/n)(1 - 1/n)^{n-1} \ge 1/(en)$. As

$f(\boldsymbol{x}) \geq (1-(1-\gamma_{\min}/k)^i) \cdot \mathrm{OPT}$, we have

$$\begin{aligned} f(\boldsymbol{x}') &\geq (1-\gamma_{\boldsymbol{x},k}/k) \cdot f(\boldsymbol{x}) + (\gamma_{\boldsymbol{x},k}/k) \cdot \mathrm{OPT} \\ &\geq (1-(1-\gamma_{\boldsymbol{x},k}/k)(1-\gamma_{\min}/k)^i) \cdot \mathrm{OPT} \\ &\geq (1-(1-\gamma_{\min}/k)^{i+1}) \cdot \mathrm{OPT}. \end{aligned}$$

Note that the last inequality holds by $\gamma_{\boldsymbol{x},k} \geq \gamma_{\min}$, because $|\boldsymbol{x}| < k$ and $\gamma_{\boldsymbol{x},k}$ decreases with $\boldsymbol{x}$. As $\boldsymbol{x}$ is mutated twice independently in line 5, such a new solution $\boldsymbol{x}'$ can be generated with probability at least $1-(1-1/(en))^2 = 2/(en)-1/(en)^2$. It is clear that $|\boldsymbol{x}'| = |\boldsymbol{x}|+1 \leq i+1$. Then, $\boldsymbol{x}'$ will be added into P; otherwise, $\boldsymbol{x}'$ must be dominated by one archived solution in line 7 of Algorithm 2, and this implies that $J_{\max}$ has been larger than i, contradicting with the assumption $J_{\max} = i$. After adding $\boldsymbol{x}'$ into P, $J_{\max} \geq i+1$. Thus, $J_{\max}$ can increase by at least 1 in one iteration with probability at least $(1/|P|^2) \cdot (2/(en) - 1/(en)^2)$. By the procedure of updating the population P in lines 6-10, the solutions in P must be incomparable. Thus, each value of one objective can correspond to at most one solution in P. Because the solutions with $|\boldsymbol{x}| \geq 2k$ have $+\infty$ value on the first objective, they must be excluded from P, and thus, $|\boldsymbol{x}| \in \{0, 1, \ldots, 2k-1\}$, implying $|P| \leq 2k$. We can now conclude that the probability of increasing $J_{\max}$ in one iteration is at least $(1/(4k^2)) \cdot (2/(en) - 1/(en)^2) = \Omega(1/(k^2 n))$.

The above analysis shows that $J_{\max}$ will not decrease, but can increase with probability $\Omega(1/(k^2n))$ in one iteration. Thus, the expected number of iterations until $J_{\max}$ increases by at least 1 is $O(k^2n)$. For $J_{\max} = k$, it requires to increase $J_{\max}$ by at most k times, implying that the expected number of iterations until finding a solution with the desired approximation guarantee is $O(k^3n)$, which is polynomial. □

Next, by an illustrative example of subset selection, we prove that PORSS can perform much better than the greedy algorithm and POSS. As presented in Definition 5, the best subset of size $(i+1)$ can be generated from the best subset of size i by adding one specific item, and the only exception is the best subset of size k, i.e., the optimal solution, which differs greatly from the best subsets of other sizes. This example represents subset selection problems where decisions have to be made in sequence to some extent.

Definition 5. *The objective function f satisfies that*

(1) $\forall 0 \leq i \leq n-1 : f(\boldsymbol{x}^i) < f(\boldsymbol{x}^{i+1})$;

(2) *if* $|\boldsymbol{x}| = i \neq k$, *then* $\forall \boldsymbol{x} \neq \boldsymbol{x}^i : f(\boldsymbol{x}) < f(\boldsymbol{x}^i)$;

(3) *if* $|\boldsymbol{x}| = k$, *then* $\forall \boldsymbol{x} \notin \{\boldsymbol{x}^*, \boldsymbol{x}^k\} : f(\boldsymbol{x}) < f(\boldsymbol{x}^k) < f(\boldsymbol{x}^*)$,

where $\boldsymbol{x}^i = 1^i 0^{n-i}$, $\boldsymbol{x}^* = 0^k 1^k 0^{n-2k}$ *and* $k \leq n/2$.

It is clear that the optimal solution is $\boldsymbol{x}^* = 0^k 1^k 0^{n-2k}$, and each $\boldsymbol{x}^i = 1^i 0^{n-i}$ is the best solution for size i except that $\boldsymbol{x}^k = 1^k 0^{n-k}$ is the runner-up for size k. Due to the greedy nature, the greedy algorithm finds $\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$ sequentially, implying that $\boldsymbol{x}^*$ cannot be found.

Lemma 2. *For subset selection with f in Definition 5, the greedy algorithm cannot find the optimal solution.*

Lemmas 3 to 5 show the expected number of iterations of POSS and PORSS until finding the optimal solution. The detailed proofs are provided in the supplementary material due to space limitations, and we introduce the proof intuition here. Both POSS and PORSS can find the solutions $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^{2k-1}\}$ efficiently. After that, the population will always consist of these solutions before finding the optimal one. Because the Hamming distance between $\boldsymbol{x}^i$ and $\boldsymbol{x}^*$ is at least k, the probability of generating the optimal solution $\boldsymbol{x}^*$ by mutation is at most $(1/n)^k$, and thus, POSS is inefficient. For PORSS, recombination from the two diverse solutions $\boldsymbol{x}^0 = 0^n$ and $\boldsymbol{x}^{2k-1} = 1^{2k-1}0^{n-2k+1}$ can generate the solution $0^k 1^{k-1} 0^{n-2k+1}$ through exchanging their first k bits, occurring with a large probability, i.e., $1/n$ or $(1/2^{2k-1}) \cdot 2$, by one-point or uniform recombination; the subsequent mutation operator can generate $\boldsymbol{x}^*$ by flipping only one specific bit, i.e., the $(2k)$-th bit, occurring with probability $(1/n)(1-1/n)^{n-1}$. Thus, PORSS can be efficient. Note that the reason for the effectiveness of recombination is consistent with that found in the last section.

Lemma 3. *For subset selection with f in Definition 5, the expected number of iterations until POSS finds the optimal solution is at least $(n/(3e^{4ek}))^k$.*

Lemma 4. *For subset selection with f in Definition 5, the expected number of iterations until PORSS with one-point recombination finds the optimal solution is at most $6ek^2n^2$.*

Lemma 5. *For subset selection with f in Definition 5, the expected number of iterations until PORSS with uniform recombination finds the optimal solution is at most $5k^2 4^k n$.*

Let $\mathbb{E}[T_m]$, $\mathbb{E}[T_o]$ and $\mathbb{E}[T_u]$ denote the expected number of iterations of POSS, PORSS with one-point and uniform recombination, respectively, for finding the optimal solution. Considering that the budget k is usually not large in real applications, we make the following observations.

Remark 1. *According to Lemmas 3 to 5, we have*

(1) *if* $k \leq O(1)$, *then* $\mathbb{E}[T_m] \geq \Omega(n^k)$,
$\mathbb{E}[T_o] \leq O(n^2)$ *and* $\mathbb{E}[T_u] \leq O(n)$;

(2) *if* $\omega(1) \leq k \leq (\log n)/20$, *then* $\mathbb{E}[T_m] \geq n^{k/4}$,
$\mathbb{E}[T_o] \leq n^2(\log n)^2$ *and* $\mathbb{E}[T_u] \leq n^{1.1}(\log n)^2$.

In other words, when k is a constant, i.e., $O(1)$, PORSS is polynomially faster than POSS, and the gap increases with k; when k continues to increase to $\omega(1)$, the gap becomes super-polynomially large.

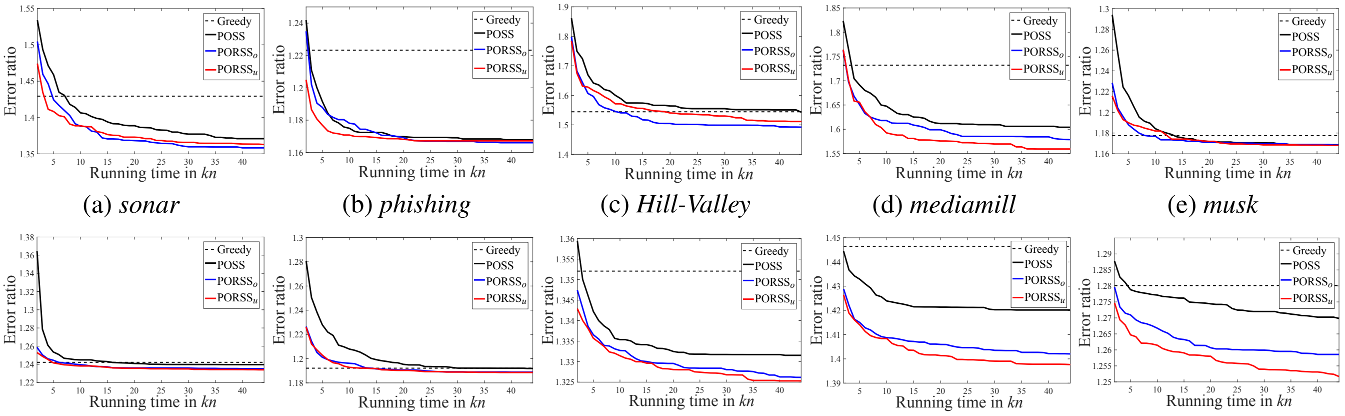
Empirical Study

In this section, we empirically compare PORSS, POSS and the greedy algorithm on the applications of unsupervised feature selection and sparse regression with various real-world data sets.[1] PORSS using one-point and uniform recombination are denoted by PORSS$_o$ and PORSS$_u$, respectively. Note that some common algorithms, e.g., IterFS (Ordozgoiti, Canaval, and Mozo 2018) for unsupervised feature selection and lasso (Tibshirani 1996) for sparse regression,

[1] https://archive.ics.uci.edu/ml/datasets.html, https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ and http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

Table 1: Unsupervised feature selection: the error ratio (the smaller, the better) of the compared algorithms on ten data sets for $k = 8$. The mean±std. is reported for randomized algorithms. In each data set, the smallest values are bolded. The count of direct win denotes the number of data sets on which POSS has a smaller error ratio 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)	OPT	Greedy	POSS	PORSS$_o$	PORSS$_u$
sonar	(208, 60)	1.353	1.429	1.371±0.007	**1.358±0.006**	1.363±0.010
phishing	(11055, 68)	1.166	1.223	1.168±0.006	**1.166±0.000**	1.167±0.003
Hill-Valley	(606, 100)	–	1.544	1.543±0.043	**1.492±0.058**	1.511±0.029
mediamill	(30993, 120)	–	1.732	1.604±0.027	1.579±0.018	**1.559±0.022**
musk	(7074, 168)	–	1.178	1.169±0.006	**1.168±0.005**	**1.168±0.005**
CT-slices	(53500, 386)	–	1.242	1.240±0.003	1.235±0.002	**1.234±0.002**
ISOLET	(7797, 617)	–	1.192	1.192±0.002	**1.189±0.001**	**1.189±0.000**
mnist	(10000, 780)	–	1.352	1.332±0.005	1.326±0.003	**1.325±0.003**
SVHN	(73257, 3072)	–	1.446	1.420±0.005	1.402±0.009	**1.398±0.005**
ORL	(400, 10304)	–	1.280	1.270±0.007	1.259±0.005	**1.252±0.004**
POSS: Count of direct win			**9.5**	-	**0**	**0**
Average rank			3.95	3.05	1.60	1.40



(a) *sonar* (b) *phishing* (c) *Hill-Valley* (d) *mediamill* (e) *musk*

(f) *CT-slices* (g) *ISOLET* (h) *mnist* (i) *SVHN* (j) *ORL*

Figure 1: Error ratio (the smaller, the better) vs. running time of POSS, PORSS$_o$ and PORSS$_u$ on unsupervised feature selection.

are not compared, because POSS has been shown to be better (Qian, Yu, and Zhou 2015; Feng, Qian, and Tang 2019).

As suggested in (Qian, Yu, and Zhou 2015), the number T of iterations of POSS is set to $2ek^2n$. Note that POSS in Algorithm 1 requires one objective evaluation for the newly generated solution $\boldsymbol{x}'$ in each iteration, whereas PORSS in Algorithm 2 needs to evaluate two new solutions $\boldsymbol{x}'', \boldsymbol{y}''$. For the fairness of comparison, the number T of iterations of PORSS is set to ek^2n; thus, the same number of objective evaluations is used. The budget k is set to 8. As POSS and PORSS are randomized algorithms, we repeat the running for ten times independently and report the average f values.

Unsupervised Feature Selection. To evaluate a submatrix $\mathbf{S}$, we measure the ratio of its reconstruction error in Definition 3 w.r.t. the smallest rank-k approximation error by SVD:

$$\text{error ratio} = \|\mathbf{A} - \mathbf{S}\mathbf{S}^{+}\mathbf{A}\|_F^2 / \|\mathbf{A} - \mathbf{A}_k\|_F^2,$$

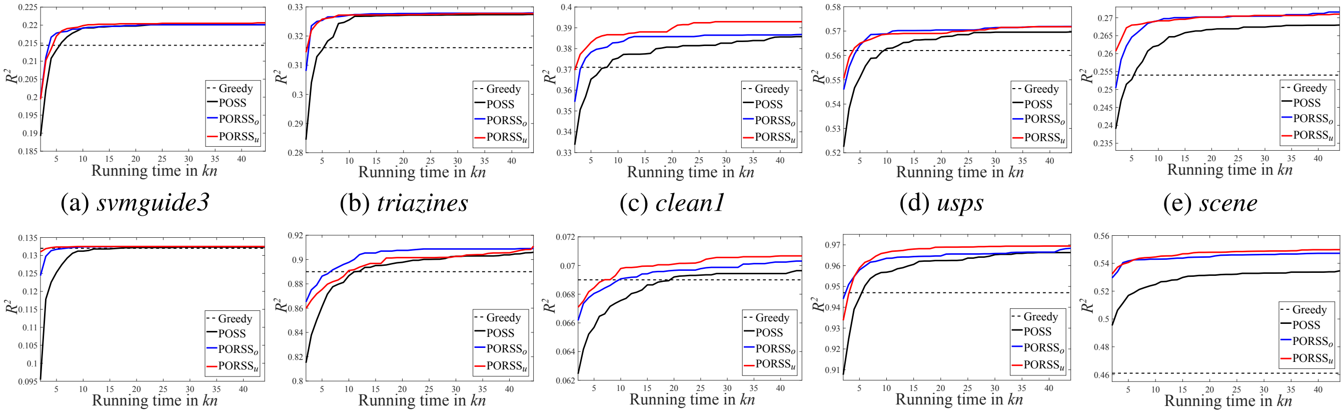
where $\mathbf{A}_k$ denotes the best rank-k approximation to $\mathbf{A}$ via SVD. The error ratio is larger than 1, and the smaller the better. The results are summarized in Table 1. Note that the standard deviation of error ratio is 0 sometimes (e.g., for PORSS$_o$ on the *phishing* data set), which is because the same good solution is found in ten runs. We can see that the best performance on each data set is always achieved by PORSS$_o$ or PORSS$_u$. By the *sign-test* (Demšar 2006) with confidence level 0.05, POSS is significantly better than the greedy algorithm, consistent with the previous results (Feng, Qian, and Tang 2019), and significantly worse than PORSS$_o$ and PORSS$_u$, showing the usefulness of recombination. The rank of each algorithm on each data set is also computed as in (Demšar 2006), and averaged in the last row of Table 1.

Sparse Regression. We use $R^2_{z,S}$ in Definition 4 to measure the goodness of a subset S of variables. The larger it is, the better. We can see from Table 2 that the algorithms have the similar performance rank as in unsupervised feature selection, i.e., PORSS$_o$ and PORSS$_u$ are significantly better than POSS, and the greedy algorithm performs the worst.

To have a more clear comparison, we select the greedy algorithm for the baseline, and plot the curve of error ratio or

Table 2: Sparse regression: the R^2 value (the larger, the better) of the compared algorithms on ten data sets for $k = 8$. The mean±std. is reported for randomized algorithms. In each data set, the largest values are bolded. The count of direct win denotes the number of data sets on which POSS 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)	OPT	Greedy	POSS	PORSS$_o$	PORSS$_u$
svmguide3	(1243, 22)	0.221	0.214	0.220±0.001	0.220±0.001	**0.221±0.001**
triazines	(186, 60)	0.328	0.316	0.327±0.000	**0.328±0.000**	**0.328±0.000**
clean1	(476, 166)	–	0.371	0.386±0.004	0.387±0.006	**0.393±0.005**
usps	(7291, 256)	–	0.562	0.570±0.003	**0.572±0.003**	**0.572±0.003**
scene	(1211, 294)	–	0.254	0.268±0.003	**0.272±0.002**	0.271±0.002
protein	(17766, 356)	–	0.132	0.132±0.000	**0.133±0.000**	**0.133±0.000**
colon-cancer	(62, 2000)	–	0.890	0.906±0.011	0.909±0.018	**0.911±0.014**
cifar10	(50000, 3072)	–	0.069	0.070±0.001	0.070±0.001	**0.071±0.001**
leukemia	(72, 7129)	–	0.947	0.966±0.009	0.968±0.006	**0.969±0.007**
smallNORB	(24300, 18432)	–	0.461	0.535±0.007	0.547±0.003	**0.550±0.002**
POSS: Count of direct win			**9.5**	–	1	0
Average rank			3.95	2.95	1.85	1.25



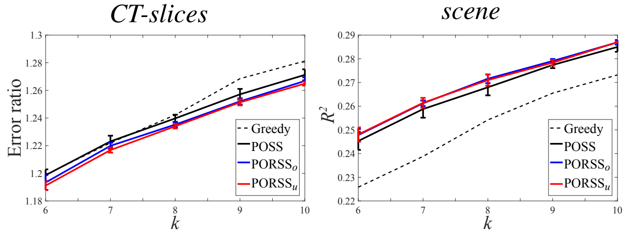
(a) *svmguide3* (b) *triazines* (c) *clean1* (d) *usps* (e) *scene*

(f) *protein* (g) *colon-cancer* (h) *cifar10* (i) *leukemia* (j) *smallNORB*

Figure 2: R^2 (the larger, the better) vs. running time of POSS, PORSS$_o$ and PORSS$_u$ on sparse regression.

R^2 over the running time for POSS, PORSS$_o$ and PORSS$_u$, as shown in Figures 1 and 2. Note that the running time is considered in the number of objective function evaluations, and one unit on the x-axis corresponds to kn evaluations, the running time of the greedy algorithm. It can be clearly observed that the curves of PORSS$_o$ and PORSS$_u$ are almost always below (above) that of POSS in Figure 1 (Figure 2), implying that PORSS$_o$ and PORSS$_u$ consistently outperform POSS during the running process. It is known that the greedy algorithm is an efficient fixed time algorithm, while PORSS is an anytime algorithm that can use more time to find better solutions. In fact, we find that PORSS can even be both better and faster, e.g., in Figures 1(h-j) and 2(j).

Note that the improvement of PORSS over POSS is small in several cases, which may be because POSS has performed very well. We compute the optimal solution by exhaustive enumeration, denoted as OPT. Due to the computation time limit, OPT is calculated only for the two smallest data sets in both applications. It can be seen from the second and third rows of Tables 1 and 2 that POSS achieves the nearly optimal solution, which also implies that PORSS can bring improvement even when POSS has been nearly optimal.



(a) Unsupervised feature selection

(b) Sparse regression

Figure 3: Comparison for budget $k \in \{6, 7, 8, 9, 10\}$.

Finally, we examine the influence of budget k in Figure 3. The results for $k \in \{6, 7, 8, 9, 10\}$ on the data set *CT-slices* for unsupervised feature selection and *scene* for sparse regression show that PORSS always performs the best.

Conclusion

This paper proposes the PORSS algorithm for subset selection, based on Pareto optimization with recombination. The superiority of PORSS over state-of-the-art algorithms, i.e., POSS and the greedy algorithm, is shown by theoretical analysis, as well as empirical study on the applications of unsupervised feature selection and sparse regression. Theoretical analysis also provides insight on the effect of recombination, which may be helpful for designing improved EAs.

References

Bäck, T. 1996. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.

Bhaskara, A.; Rostamizadeh, A.; Altschuler, J.; Zadimoghaddam, M.; Fu, T.; and Mirrokni, V. 2016. Greedy column subset selection: New bounds and distributed algorithms. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*, 2539–2548.

Dang, D.-C.; Friedrich, T.; Kötzing, T.; Krejca, M.; Lehre, P. K.; Oliveto, P. S.; Sudholt, D.; and Sutton, A. 2018. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation* 22(3):484–497.

Das, A., and Kempe, D. 2011. 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)*, 1057–1064.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2):182–197.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7:1–30.

Doerr, B.; Johannsen, D.; Kötzing, T.; Neumann, F.; and Theile, M. 2013. More effective crossover operators for the all-pairs shortest path problem. *Theoretical Computer Science* 471:12–26.

Farahat, A. K.; Ghodsi, A.; and Kamel, M. S. 2011. An efficient greedy method for unsupervised feature selection. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'11)*, 161–170.

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

Feng, C.; Qian, C.; and Tang, K. 2019. Unsupervised feature selection by Pareto optimization. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, 3534–3541.

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

Giel, O. 2003. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'03)*, 1918–1925.

Harshaw, C.; Feldman, M.; Ward, J.; and Karbasi, A. 2019. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, 2634–2643.

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

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. 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)*, 137–146.

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

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

Lin, H., and Bilmes, J. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL'11)*, 510–520.

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

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

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

Neumann, F., and Theile, M. 2010. How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN'10)*, 667–676.

Oliveto, P. S., and Witt, C. 2014. On the runtime analysis of the simple genetic algorithm. *Theoretical Computer Science* 545:2–19.

Ordozgoiti, B.; Canaval, S.; and Mozo, A. 2018. Iterative column subset selection. *Knowledge and Information Systems* 54(1):65–94.

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

Qian, C.; Shi, J.-C.; Yu, Y.; Tang, K.; and Zhou, Z.-H. 2017. Subset selection under noise. In *Advances in Neural Information Processing Systems 30 (NIPS'17)*, 3562–3572.

Qian, C.; Li, G.; Feng, C.; and Tang, K. 2018. Distributed Pareto optimization for subset selection. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, 1492–1498.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2013. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence* 204:99–119.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2015. Subset selection by Pareto optimization. In *Advances in Neural Information Processing Systems 28 (NIPS'15)*, 1765–1773.

Qian, C. 2019. Distributed Pareto optimization for large-scale noisy subset selection. *IEEE Transactions on Evolutionary Computation* in press.

Roostapour, V.; Neumann, A.; Neumann, F.; and Friedrich, T. 2019. Pareto optimization for subset selection with dynamic cost constraints. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, 2354–2361.

Sudholt, D. 2017. How crossover speeds up building block assembly in genetic algorithms. *Evolutionary Computation* 25(2):237–274.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58(1):267–288.

Supplementary Material: Subset Selection by Pareto Optimization with Recombination

Chao Qian,[1*] **Chao Bian,**[1] **Chao Feng**[2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
qianc@lamda.nju.edu.cn, chaobian12@gmail.com, chaofeng@mail.ustc.edu.cn

First, we provide the detailed analysis for $Q_m(d,j)$, i.e., (c.1) and (c.2) in the subsection of analyzing the influence of recombination, which is omitted in our original paper due to space limitations.

The main idea is to show that there exists $0 < j_0 \le d$ such that $q_{0,j}/q_{d,j} > 1$ for $j < j_0$ and $q_{0,j}/q_{d,j} \le 1$ for $j \ge j_0$. We first give some bounds on the probability $q_{i,j}$ that will be used. If $i \ge j$, we have

$$\binom{i}{i-j}\left(\frac{1}{n}\right)^{i-j}\left(1-\frac{1}{n}\right)^{n-i+j} \le q_{i,j} \le \binom{i}{i-j}\left(\frac{1}{n}\right)^{i-j},$$

where the first inequality holds because to generate a solution with j different bits from $\boldsymbol{x}$ by mutating a solution with i different bits from $\boldsymbol{x}$, it is sufficient to flip $(i-j)$ different bits and keep other bits unchanged; the second inequality holds because it requires to flip at least $(i-j)$ different bits. $\forall 0 \le j \le n$, we have $q_{0,j} = \binom{n}{j}(1/n)^j(1-1/n)^{n-j}$, because exactly j bits need to be flipped.

For $0 \le j \le d-1$, we have

$$\begin{aligned}\frac{q_{0,j+1}}{q_{0,j}} &= \frac{\binom{n}{j+1}(\frac{1}{n})^{j+1}(1-\frac{1}{n})^{n-j-1}}{\binom{n}{j}(\frac{1}{n})^{j}(1-\frac{1}{n})^{n-j}}\\ &= \frac{n-j}{j+1}\cdot\frac{1}{n(1-1/n)} = \frac{n-j}{(j+1)(n-1)},\end{aligned}$$

and

$$\begin{aligned}\frac{q_{d,j}}{q_{d,j+1}} &\le \frac{\binom{d}{d-j}(\frac{1}{n})^{d-j}}{\binom{d}{d-j-1}(\frac{1}{n})^{d-j-1}(1-\frac{1}{n})^{n-d+j+1}}\\ &\le \frac{2e(j+1)}{(d-j)n},\end{aligned}$$

where the last inequality holds by $(1-1/n)^{n-d+j+1} \ge (1-1/n)^n \ge 1/e\cdot(1-1/n) \ge 1/(2e)$. Thus, we have

$$\begin{aligned}\frac{q_{0,j+1}}{q_{d,j+1}}\cdot\frac{q_{d,j}}{q_{0,j}} &\le \frac{n-j}{(j+1)(n-1)}\cdot\frac{2e(j+1)}{(d-j)n}\\ &= \frac{2e(n-j)}{(d-j)(n-1)n}\\ &\le \frac{2e(n-d+1)}{(n-1)n} < 1,\end{aligned}$$

where the second inequality holds because $(n-j)/(d-j) = (n-d+d-j)/(d-j) = 1+(n-d)/(d-j) \le n-d+1$, and the last inequality holds with $n \ge 8$. Thus, $q_{0,j}/q_{d,j}$ decreases with j. Note that $q_{d,0} = (1/n)^d(1-1/n)^{n-d}$ because exactly d bits need to be flipped. Thus, we have

$$\frac{q_{0,0}}{q_{d,0}} = \frac{(1-\frac{1}{n})^n}{(\frac{1}{n})^d(1-\frac{1}{n})^{n-d}} = (n-1)^d > 1. \tag{1}$$

Meanwhile, we have

$$\begin{aligned}\frac{q_{0,d}}{q_{d,d}} &\le \frac{\binom{n}{d}(\frac{1}{n})^d(1-\frac{1}{n})^{n-d}}{(1-\frac{1}{n})^n}\\ &\le \left(\frac{e}{d}\right)^d\left(1-\frac{1}{n}\right)^{-d} < 1,\end{aligned} \tag{2}$$

where the second inequality holds by $\binom{n}{d} \le (en/d)^d$, and the last inequality holds for $d \ge 3$. According to Eqs. (1), (2) and using the fact that $q_{0,j}/q_{d,j}$ decreases with j, there must exist $0 < j_0 \le d$ such that $q_{0,j}/q_{d,j} > 1$ for $j < j_0$ and $q_{0,j}/q_{d,j} \le 1$ for $j \ge j_0$.

For (c.1), consider $j < j_0$. We have

$$Q_m(d,j) \le q_{0,j}+q_{d,j} < 2q_{0,j} \le 2\binom{n}{j}\left(\frac{1}{n}\right)^j \le 2\left(\frac{e}{j}\right)^j,$$

and

$$Q_m(d,j) \ge q_{0,j} = \binom{n}{j}\left(\frac{1}{n}\right)^j\left(1-\frac{1}{n}\right)^{n-j} \ge \frac{1}{2ej^j},$$

where the last inequality holds by $\binom{n}{j} \ge (n/j)^j$.

For (c.2), consider $j \ge j_0$. We have

$$\begin{aligned}Q_m(d,j) \le q_{0,j}+q_{d,j} \le 2q_{d,j} &\le 2\binom{d}{d-j}\left(\frac{1}{n}\right)^{d-j}\\ &\le 2\left(\frac{ed}{d-j}\right)^{d-j}\left(\frac{1}{n}\right)^{d-j}\\ &= 2\left(\frac{e}{d-j}\cdot\frac{d}{n}\right)^{d-j},\end{aligned}$$

and

[*]This work was supported by the NSFC (61603367).

$$\begin{aligned} Q_m(d,j) \ge q_{d,j} &\ge \binom{d}{d-j}\left(\frac{1}{n}\right)^{d-j}\left(1-\frac{1}{n}\right)^{n-d+j} \\ &\ge \left(\frac{d}{d-j}\right)^{d-j}\cdot\left(\frac{1}{n}\right)^{d-j}\cdot\frac{1}{2e} \\ &= \frac{1}{2e}\cdot\left(\frac{1}{d-j}\cdot\frac{d}{n}\right)^{d-j}. \end{aligned}$$

Next, we provide the detailed proofs of Lemmas 3 to 5.

Proof of Lemma 3. Consider a phase of $2ek(2k-1)n$ iterations since POSS starts. Because the initial solution is 0^n and the optimal solution is $\boldsymbol{x}^* = 0^k1^k0^{n-2k}$, it requires to flip the $(2k)$-th bit at least once in this phase for finding $\boldsymbol{x}^*$. In each iteration, each bit of a solution selected from the population P is flipped independently with probability $1/n$ by bit-wise mutation. Thus, the probability that the $(2k)$-th bit is never flipped in these $2ek(2k-1)n$ iterations, implying that the optimal solution $\boldsymbol{x}^*$ is not found in this phase, is

$$\left(1-\frac{1}{n}\right)^{2ek(2k-1)n} \ge \left(1-\frac{1}{n}\right)^{4ek^2(n-1)} \ge \left(\frac{1}{e}\right)^{4ek^2},$$

where the first inequality holds by $k \le n/2$.

Under the above condition, we next examine the probability of finding the solutions $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$ in these $2ek(2k-1)n$ iterations. Considering that $\boldsymbol{x}^*$ is not found, $\forall 0 \le i \le 2k-1$, $\boldsymbol{x}^i$ will always be maintained in P once found, because for any solution $\boldsymbol{x}$ with $|\boldsymbol{x}| = j < i$, $f(\boldsymbol{x}) \le f(\boldsymbol{x}^j) < f(\boldsymbol{x}^i)$, and thus, $\boldsymbol{x}^i$ cannot be dominated. Now we show that POSS can find $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$ by following the path $\boldsymbol{x}^0 \to \boldsymbol{x}^1 \to \dots \to \boldsymbol{x}^{2k-1}$. Let $P_{\max}$ denote the largest size of P during the running of POSS. Initially, $\boldsymbol{x}^0 \in P$. After $\boldsymbol{x}^{i-1}$ has been found, $\boldsymbol{x}^i$ can be generated by flipping only the i-th bit of $\boldsymbol{x}^{i-1}$, occurring with probability at least $(1/P_{\max}) \cdot (1/n)(1-1/n)^{n-1} \ge 1/(enP_{\max})$, where $1/P_{\max}$ is a lower bound on the probability of selecting $\boldsymbol{x}^{i-1}$ in line 3 of POSS due to uniform selection, and $(1/n)(1-1/n)^{n-1}$ is the probability of flipping only one specific bit by mutation. Thus, after $\boldsymbol{x}^{i-1}$ has been found, $\boldsymbol{x}^i$ can be found in $2ekn$ iterations with probability at least $1-(1-1/(enP_{\max}))^{2ekn}$. We divide the $2ek(2k-1)n$ iterations into $(2k-1)$ subphases with equal length, where each subphase with length $2ekn$ corresponds to one "$\to$" on the path $\boldsymbol{x}^0 \to \boldsymbol{x}^1 \to \dots \to \boldsymbol{x}^{2k-1}$. Thus, under the condition that $\boldsymbol{x}^*$ is not found, $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$ can be found in $2ek(2k-1)n$ iterations with probability at least

$$\left(1-\left(1-\frac{1}{enP_{\max}}\right)^{2ekn}\right)^{2k-1} \ge \left(1-\left(\frac{1}{e}\right)^{\frac{2k}{P_{\max}}}\right)^{2k-1}.$$

According to the analysis above, in the first $2ek(2k-1)n$ iterations of POSS, the probability that $\boldsymbol{x}^*$ is not found while $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$ are found is at least

$$\left(\frac{1}{e}\right)^{4ek^2}\cdot\left(1-\left(\frac{1}{e}\right)^{\frac{2k}{P_{\max}}}\right)^{2k-1} \ge \left(\frac{1}{e}\right)^{4ek^2}\left(1-\frac{1}{e}\right)^{2k}.$$

Note that $P_{\max} \le 2k$, as derived in the proof of Theorem 1.

For any solution $\boldsymbol{x} \notin \{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}, \boldsymbol{x}^*\}$ with $|\boldsymbol{x}| = j$, it must be dominated by some solution in $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$, because if $j \ge 2k$, we have $f_1(\boldsymbol{x}) = +\infty$; otherwise, $f(\boldsymbol{x}) < f(\boldsymbol{x}^j)$. Thus, after $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$ have been found, P will always consist of $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$ until finding $\boldsymbol{x}^*$. It can be verified that $\forall 0 \le i \le 2k-1 : H(\boldsymbol{x}^i, \boldsymbol{x}^*) \ge k$. Thus, the probability of generating $\boldsymbol{x}^*$ by mutation in one iteration is at most $(1/n)^k$, implying that the expected number of iterations for finding $\boldsymbol{x}^*$ is at least n^k.

Thus, the expected number of iterations until POSS finds the optimal solution $\boldsymbol{x}^*$ is at least

$$\left(\frac{1}{e}\right)^{4ek^2}\left(1-\frac{1}{e}\right)^{2k}\cdot n^k \ge \left(\frac{n}{3e^{4ek}}\right)^k,$$

where the inequality holds by $(1-1/e)^2 \ge 1/3$. □

Proof of Lemma 4. We divide the optimization procedure into two phases: (1) starts after initialization and finishes until finding $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$; (2) starts after phase (1) and finishes until finding the optimal solution $\boldsymbol{x}^*$.

For phase (1), we can pessimistically assume that $\boldsymbol{x}^*$ is not found, because the goal is to derive an upper bound on the number of iterations for finding $\boldsymbol{x}^*$. From the proof of Lemma 3, it is known that $\forall 0 \le i \le 2k-1$, $\boldsymbol{x}^i$ will always be maintained in the population P once found. We then analyze the probability of generating $\boldsymbol{x}^i$ in one iteration after finding $\boldsymbol{x}^{i-1}$. Consider that the two solutions selected in line 3 of PORSS are both $\boldsymbol{x}^{i-1}$, i.e., $\boldsymbol{x} = \boldsymbol{y} = \boldsymbol{x}^{i-1}$, occurring with probability $1/|P|^2$ due to uniform selection with replacement. For two identical solutions, recombination will not make any change, i.e., the two solutions generated by recombination in line 4 are $\boldsymbol{x}' = \boldsymbol{y}' = \boldsymbol{x}^{i-1}$. Then, $\boldsymbol{x}^i$ can be generated by the subsequent mutation operator, through flipping only the i-th bit of $\boldsymbol{x}' = \boldsymbol{x}^{i-1}$, with probability $1/n \cdot (1-1/n)^{n-1} \ge 1/(en)$. As the analysis in the proof of Theorem 1, we have $|P| \le 2k$. Thus, the probability of generating $\boldsymbol{x}^i$ in one iteration after finding $\boldsymbol{x}^{i-1}$ is at least $(1/|P|^2) \cdot (1/(en)) \ge 1/(4ek^2n)$, implying that the expected number of iterations of phase (1), i.e., for finding $\{\boldsymbol{x}^0, \boldsymbol{x}^1, \dots \boldsymbol{x}^{2k-1}\}$, is at most $4ek^2n \cdot (2k-1)$.

Now we examine phase (2). Consider that the two solutions $\boldsymbol{x}^0$ and $\boldsymbol{x}^{2k-1}$ are selected in line 3 of PORSS for recombination, occurring with probability $2/|P|^2 \ge 1/(2k^2)$. Then, the solution $0^k1^{k-1}0^{n-2k+1}$ can be generated by one-point recombination, through exchanging the first k bits of $\boldsymbol{x}^0$ and $\boldsymbol{x}^{2k-1}$, with probability $1/n$. The subsequent mutation operator can generate $\boldsymbol{x}^*$ by flipping the $(2k)$-th bit of $0^k1^{k-1}0^{n-2k+1}$, occurring with probability $1/n \cdot (1-1/n)^{n-1} \ge 1/(en)$. Thus, the probability of generating $\boldsymbol{x}^*$ in one iteration is at least $(1/(2k^2)) \cdot (1/n) \cdot (1/(en))$, implying that the expected number of iterations of phase (2), i.e., for finding $\boldsymbol{x}^*$, is at most $2ek^2n^2$.

Combining phases (1) and (2), the total expected number of iterations for finding the optimal solution is at most

$$4ek^2(2k-1)n + 2ek^2n^2 \le 6ek^2n^2,$$

where the inequality holds by $k \le n/2$. □

Proof of Lemma 5. The proof is similar to that of Lemma 4. The only difference is the probability of generating the solution $0^k1^{k-1}0^{n-2k+1}$ by recombining $\boldsymbol{x}^0 = 0^n$ and $\boldsymbol{x}^{2k-1} = 1^{2k-1}0^{n-2k+1}$ in phase (2), led by the different recombination operators. By uniform recombination, $0^k1^{k-1}0^{n-2k+1}$ can be generated by exchanging the first k bits, or the bits from positions $(k+1)$ to $(2k-1)$, of $\boldsymbol{x}^0$ and $\boldsymbol{x}^{2k-1}$, occurring with probability $(1/2)^{2k-1} \cdot 2$. Thus, the probability of generating $\boldsymbol{x}^*$ in one iteration is at least $(1/(2k^2)) \cdot (1/2)^{2k-2} \cdot (1/(en))$, implying that the expected number of iterations of phase (2) is at most $ek^2 2^{2k-1} n$. Combining the expected number of iterations, i.e., at most $4ek^2(2k-1)n$, of phase (1), the total expected number of iterations is at most

$$ek^2n(4(2k-1) + 2^{2k-1}) \leq ek^2n(2^{2k} + 2^{2k-1}) \leq 5k^2 4^k n.$$

$\square$