Maximizing Submodular or Monotone Approximately Submodular Functions by Multi-objective Evolutionary Algorithms

Chao Qian[1], Yang Yu[1], Ke Tang[2], Xin Yao[2], Zhi-Hua Zhou[1]*

[1] *National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China*

[2] *Shenzhen Key Laboratory of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China*

Abstract

Evolutionary algorithms (EAs) are a kind of nature-inspired general-purpose optimization algorithm, and have shown empirically good performance in solving various real-word optimization problems. During the past two decades, promising results on the running time analysis (one essential theoretical aspect) of EAs have been obtained, while most of them focused on isolated combinatorial optimization problems, which do not reflect the general-purpose nature of EAs. To provide a general theoretical explanation of the behavior of EAs, it is desirable to study their performance on general classes of combinatorial optimization problems. To the best of our knowledge, the only result towards this direction is the provably good approximation guarantees of EAs for the problem class of maximizing monotone submodular functions with matroid constraints. The aim of this work is to contribute to this line of research. Considering that many combinatorial optimization problems involve non-monotone or non-submodular objective functions, we study the general problem classes, maximizing submodular functions with/without a size constraint and maximizing monotone approximately submodular functions with a size constraint. We prove that a simple multi-objective EA called GSEMO-C can generally achieve good approximation guarantees in polynomial expected running time.

Key words: Evolutionary algorithms, submodular optimization, multi-objective evolutionary algorithms, running time analysis, computational complexity

*Corresponding author

Email addresses: qianc@lamda.nju.edu.cn (Chao Qian[1]), yuy@lamda.nju.edu.cn (Yang Yu[1]), tangk3@sustech.edu.cn (Ke Tang[2]), xiny@sustech.edu.cn (Xin Yao[2]), zhouzh@lamda.nju.edu.cn (Zhi-Hua Zhou[1])

1. Introduction

Evolutionary algorithms (EAs) [3] are a kind of randomized metaheuristic optimization algorithm, inspired by the evolution process of natural species, i.e., natural selection and survival of the fittest. Starting from a random population of solutions, EAs iteratively apply reproduction (e.g., mutation and recombination) operators to generate offspring solutions from the current population, and then apply a selection operator to eliminate less desirable solutions. EAs have been applied to diverse areas (e.g., robotics [32], networks [47] and machine learning [48]) and can produce human-competitive results [25]. Compared with the application, the theoretical analysis of EAs is, however, far behind. Many researchers thus have been devoted to understanding the behavior of EAs from a theoretical point of view, which is still an ongoing challenge.

During the past two decades, a lot of progress has been made on the running time analysis of EAs, which is one essential theoretical aspect. The running time measures how many objective (i.e., fitness) function evaluations an EA needs until finding an optimal solution or an approximate solution. The running time analysis of EAs started with artificial example problems. In [9, 10], a simple single-objective EA called (1+1)-EA has been shown to be able to solve two well-structured pseudo-Boolean problems OneMax and LeadingOnes in $\Theta(n \log n)$ and $\Theta(n^2)$ (where n is the problem size) expected running time, respectively. These two problems are to maximize the number of 1-bits of a solution and the number of consecutive 1-bits counting from the left of a solution, respectively. Both of them have a short path with increasing fitness to the optimum. For some problems (e.g., SPC) where there is a short path with constant fitness to the optimum, the (1+1)-EA can also find an optimal solution in polynomial expected time [22]. But when the problem (e.g., Trap) has a deceptive path, i.e., a path with increasing fitness away from the optimum, the (1+1)-EA will need exponential running time [20]. More results can be found in [2].

The analysis on simple artificial problems disclosed theoretical properties of EAs (e.g., which problem structures are easy or hard for EAs), and also helped to develop approaches for analyzing more complex problems. The running time analysis of EAs was then extended to combinatorial optimization problems. For some P-solvable problems, EAs have been shown to be able to find an optimal solution in polynomial expected time. For example, the minimum spanning tree problem can be solved by the (1+1)-EA and a simple multi-objective EA called GSEMO in $O(m^2(\log n + \log w_{\max}))$ [37] and $O(mn(n + \log w_{\max}))$ [36] expected time, respectively. Note that m, n and $w_{\max}$ are the number of edges, the number of nodes and the maximum edge weight of a graph, respectively. For some NP-hard problems, EAs have been shown to be able to achieve good approximation ratios in polynomial expected time. For example, for the partition problem, the (1+1)-EA can achieve a $(4/3)$-approximation ratio in $O(n^2)$ expected time [46]; for the minimum set cover problem, the expected

running time of the GSEMO until obtaining a $(\log m+1)$-approximation ratio is $O(m^2n+mn(\log n+\log c_{\max}))$ [14], where m, n and $c_{\max}$ denote the size of the ground set, the number of subsets and the maximum cost of a subset, respectively. For more running time results of EAs on combinatorial optimization problems, the reader can refer to [38].

For the analysis of the GSEMO (which is a multi-objective EA) on single-objective optimization problems (e.g., minimum spanning tree and minimum set cover), the original single-objective problem is transformed into a multi-objective problem, which is then solved by the GSEMO. Note that multi-objective optimization here is just an intermediate process, which might be beneficial [14, 36, 39, 41], and we still focus on the quality of the best solution w.r.t. the original single-objective problem, in the population found by the GSEMO. Running time analysis of EAs on real multi-objective optimization problems has also been investigated, where the running time is measured by the number of fitness evaluations until finding the Pareto front (which represents different optimal tradeoffs between the multiple objectives) or an approximation of the Pareto front. For example, Giel [17] proved that the GSEMO can solve the bi-objective pseudo-Boolean problem LOTZ in $O(n^3)$ expected time; for the NP-hard bi-objective minimum spanning tree problem, it has been shown that the GSEMO can obtain a 2-approximation ratio in pseudo-polynomial time [35, 40].

The analysis on combinatorial optimization problems helped to reveal the ability of EAs. However, most of the previous promising results were obtained for isolated problems, while EAs are known to be general-purpose optimization algorithms, which can be applied to various problems. Thus, it is more desirable to provide a general theoretical explanation of the behavior of EAs, that is, to theoretically study the performance of EAs on general classes of combinatorial optimization problems.

To the best of our knowledge, only two pieces of work in this direction have been reported. Reichel and Skutella [44] first studied the problem class of maximizing linear functions with k matroid constraints, which includes some well-known combinatorial optimization problems such as maximum matching, Hamiltonian path, etc. They proved that the (1+1)-EA can obtain a $(1/k)$-approximation ratio in $O(n^{k+2}(\log r+\log w_{\max}))$ expected running time, where n, r and $w_{\max}$ denote the size of the ground set, the minimum rank of the ground set w.r.t. one matroid and the maximum weight of an element, respectively. Later, Friedrich and Neumann [13] considered a more general problem class, where the objective function is relaxed to satisfy the monotone and submodular property. The (1+1)-EA has been shown to be able to achieve a $(\frac{1}{k+1/p+\epsilon})$-approximation ratio in $O(\frac{1}{\epsilon}n^{2p(k+1)+1}k\log n)$ expected time, where $p\geq 1$ and $\epsilon>0$. They also studied a specific non-monotone case, i.e., symmetric objective functions, and proved that the expected running time until the GSEMO obtains a $(\frac{1}{(k+2)(1+\epsilon)})$-approximation ratio for maximizing symmetric submodular functions with k matroid constraints is $O(\frac{1}{\epsilon}n^{k+6}\log n)$.

The aim of this paper is to contribute to this line of research. Considering that the objective function of many combinatorial optimization problems can be non-monotone (not necessarily symmetric) or non-submodular, we study the performance of EAs on the general problem classes, maximizing submodular functions with/without a size constraint and maximizing monotone approximately submodular functions with a size constraint. Note that the objective function is a set function $f : 2^V \to \mathbb{R}$ which maps a subset of the ground set V to a real value, and a size constraint means that the size of a subset is no larger than a budget k. We prove that for any concerned problem class, a variant of the GSEMO, called GSEMO-C, can obtain a good approximation guarantee in polynomial expected running time. Our main results can be summarized as follows.

- For the problem class of maximizing non-monotone submodular functions without constraints, with special instances including maximum cut [18], maximum facility location [1] and variants of the maximum satisfiability problem [19], we prove that the GSEMO-C achieves a constant approximation ratio of $(\frac{1}{3} - \frac{\epsilon}{n})$ in $O(\frac{1}{\epsilon}n^4 \log n)$ expected running time (i.e., **Theorem 1**), where n is the size of the ground set V and $\epsilon > 0$.

- For the problem class of maximizing submodular and approximately monotone functions with a size constraint, with special instances such as sensor placement [28], we prove that the GSEMO-C within $O(n^2(\log n + k))$ expected time finds a subset X with $f(X) \geq (1 - 1/e) \cdot (\mathrm{OPT} - k\epsilon)$ (i.e., **Theorem 2**), where e is the base of the natural logarithm, OPT denotes the optimal function value, and $\epsilon \geq 0$ captures the degree of approximate monotonicity.

- For the problem class of maximizing monotone and approximately submodular functions with a size constraint, with special instances including sparse regression [8], dictionary selection [26] and Bayesian experimental design [28], we prove the approximation guarantee of the GSEMO-C w.r.t. each notion of "approximate submodularity", which measures how close a general set function f is to submodularity.
  (1) In [26], a set function f is ϵ-approximately submodular if the diminishing returns property holds with some deviation $\epsilon \geq 0$, i.e., for any $X \subseteq Y \subseteq V$ and $v \notin Y$, $f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y) - \epsilon$. f is submodular iff $\epsilon = 0$. We prove that the GSEMO-C within $O(n^2(\log n + k))$ expected time finds a subset X with $f(X) \geq (1 - 1/e) \cdot (\mathrm{OPT} - k\epsilon)$ (i.e., **Theorem 3**).
  (2) In [8], the approximately submodular degree of a set function f is characterized by a quantity γ called submodularity ratio. f is submodular iff $\gamma = 1$. We prove that the GSEMO-C within $O(n^2(\log n + k))$ expected time finds a subset X with $f(X) \geq (1 - e^{-\gamma}) \cdot \mathrm{OPT}$ (i.e., **Theorem 4**).
  (3) In [21], a set function f is ϵ-approximately submodular if there exists a submodular set function g such that $\forall X \subseteq V$, $(1 - \epsilon)g(X) \leq f(X) \leq (1 + \epsilon)g(X)$. f is submodular iff $\epsilon = 0$. We prove that the GSEMO-C within $O(n^2(\log n + k))$ expected time finds a subset X with $f(X) \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1 - e^{-1}(\frac{1-\epsilon}{1+\epsilon})^k) \cdot \mathrm{OPT}$ (i.e., **Theorem 5**).

Because EAs are general-purpose algorithms which utilize a small amount of problem knowledge, we cannot expect them to beat the best problem-specific algorithm. For maximizing non-monotone submodular functions without constraints, the approximation ratio of nearly $1/3$ obtained by the GSEMO-C is worse than the best known one $1/2$, which was previously obtained by the double greedy algorithm [7]. For maximizing submodular and approximately monotone, or monotone and approximately submodular, functions with a size constraint, the approximation guarantees obtained by the GSEMO-C always reach the best known ones, which were previously obtained by the standard greedy algorithm [8, 21, 26, 28]. Note that the approximate guarantees here are achieved within polynomial time, while the GSEMO-C is actually an anytime algorithm and can find better solutions by running longer. If the running time is allowed to be infinite, the GSEMO-C can eventually find an optimal solution, since the mutation operator employed for reproduction is a global search operator leading to a positive probability of generating any solution in each iteration.

Friedrich and Neumann [13] have proved that for maximizing monotone submodular functions with a size constraint, the GSEMO can achieve the approximation ratio of $(1-1/e)$, which is optimal in general [33]. Without further assumptions or knowledge of the function, no polynomial time algorithm can provide a better approximation guarantee unless P=NP. Note that their result is generalized by our analysis for submodular and approximately monotone, or monotone and approximately submodular, functions. When the function is monotone submodular, the parameters characterizing the approximately monotone or submodular degree satisfy that $\epsilon = 0$ and $\gamma = 1$, and the approximation guarantees in Theorems 2-5 all specialize to $1 - 1/e$, consistent with [13]. Furthermore, our analysis may provide guidance under what conditions the GSEMO-C can have bounded approximation guarantees, even when the function is non-monotone or non-submodular. We have shown that the performance of the GSEMO-C is theoretically guaranteed for diverse applications with non-monotone or non-submodular objective functions, including sensor placement [28], sparse regression [8], sparse support selection [11], dictionary selection [26], Bayesian experimental design [28] and determinantal function maximization [4] (i.e., **Corollaries 1- 6**). Our analytical results on general problem classes together with the previous ones [13, 44] provide a theoretical explanation for the empirically good behaviors of EAs in diverse applications.

The rest of this paper is organized as follows. Sections 2 and 3 introduce the concerned problem classes and algorithm, respectively. Section 4 presents the analysis for submodular function maximization with/without a size constraint. Section 5 presents the analysis for monotone approximately submodular function maximization with a size constraint. Section 6 concludes the paper.

2. Problem Classes

In this section, we introduce the problem classes studied in this paper. Let $\mathbb{R}$ and $\mathbb{R}^+$ denote the set of reals and non-negative reals, respectively. Given a finite non-empty set $V = \{v_1, v_2, \ldots, v_n\}$, we study the functions $f : 2^V \to \mathbb{R}$ defined on subsets of V. A set function $f : 2^V \to \mathbb{R}$ is monotone if for any $X \subseteq Y$, $f(X) \leq f(Y)$, which implies that adding more elements to a set never decreases the function value. Without loss of generality, we assume that monotone functions are normalized, i.e., $f(\emptyset) = 0$. A set function f is submodular [34] if for any $X \subseteq Y \subseteq V$ and $v \notin Y$,

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

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

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

Eq. (1) intuitively represents the "diminishing returns" property, i.e., adding an element to a set X gives a larger benefit than adding the same element to a superset Y of X. Eq. (2) implies that the benefit by adding a set of elements to a set X is smaller than the combined benefits of adding its individual elements to X. We assume that a set function f is given by a value oracle, i.e., for a given subset X, an algorithm can query an oracle to obtain the value $f(X)$. In the following, let OPT denote the optimal function value.

2.1. Submodular Function Maximization with/without a Size Constraint

We consider the problem class of submodular function maximization, where the objective function is submodular, but not necessarily monotone. Both the situations without constraints as well as with a size constraint will be studied. Without loss of generality, we assume that the objective function f is non-negative.

Definition 1 (Non-monotone Submodular Function Maximization without Constraints)
Given a non-monotone and submodular function $f : 2^V \to \mathbb{R}^+$, to find a subset $X \subseteq V$ such that

$$\arg\max_{X \subseteq V} \quad f(X).$$

For the problem without constraints as presented in Definition 1, the goal is to maximize a non-monotone submodular set function. The best known approximation guarantee is $1/2$, which was achieved by the double greedy algorithm [7]. This problem generalizes many NP-hard combinatorial optimization problems, e.g., maximum cut [18]. Let $G = (V, E)$ be a graph with non-negative edge weights $w : E \to \mathbb{R}^+$, where V and E are the set of nodes and edges, respectively. For a subset X

of nodes, let $c(X)$ be the set of edges whose nodes are in X and $V \setminus X$, respectively. The maximum cut problem is to find a subset X of nodes maximizing the weighted cut function $\sum_{e \in c(X)} w(e)$, which is submodular but not monotone. More examples include maximum facility location [1], variants of the maximum satisfiability problem [19], etc.

Definition 2 (Approximately Monotone Submodular Function Maximization with a Size Constraint) *Given a submodular and approximately monotone function $f : 2^V \to \mathbb{R}^+$ and a budget k, to find a subset $X \subseteq V$ such that*

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

For the problem with a size constraint as presented in Definition 2, the objective function, though not monotone, is required to be approximately monotone. The notion of "approximate monotonicity" [28] was introduced to measure to what extent a general set function f has the monotone property. As presented in Definition 3, a set function is ϵ-approximately monotone implies that adding one element to a set decreases the function by at most ϵ.

Definition 3 (ϵ-Approximate Monotonicity [28])
Let $\epsilon \geq 0$. A set function $f : 2^V \to \mathbb{R}$ is ϵ-approximately monotone if for any $X \subseteq V$ and $v \notin X$,

$$f(X \cup \{v\}) \geq f(X) - \epsilon.$$

It is easy to see that f is monotone iff $\epsilon = 0$. The standard greedy algorithm, which iteratively adds one element with the largest f improvement until k elements are selected, has been proved to achieve a subset X with $f(X) \geq (1 - 1/e) \cdot (\mathrm{OPT} - k\epsilon)$ [28]. A typical application is the sensor placement task, i.e., to select locations to install a limited number of sensors such that spatial phenomena can be monitored well. A common criterion to be maximized is the mutual information, which is submodular but not monotone. It has been shown [28] that a polynomial discretization level of locations can guarantee that the mutual information is ϵ-approximately monotone. Note that there are applications (e.g., maximum entropy sampling [45]) where the objective function is even not approximately monotone, i.e., ϵ is not well bounded.

2.2. Monotone Approximately Submodular Function Maximization with a Size Constraint

Another concerned problem class is presented in Definition 4. The goal is to find a subset with at most k elements such that a given monotone and approximately submodular set function is maximized. Note that the situation without constraints is not considered, as it is trivial that an optimal solution is the whole set V for monotone functions.

Definition 4 (Monotone Approximately Submodular Function Maximization with a Size Constraint)
Given a monotone and approximately submodular function $f : 2^V \to \mathbb{R}^+$ and a budget k, to find a subset $X \subseteq V$ such that

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

Several notions of "approximate submodularity" [8, 21, 26] were introduced to measure to what extent a set function f has the submodular property. For each approximately submodular notion, the best known approximation guarantee was achieved by the standard greedy algorithm [8, 21, 26].

In [26], the approximate submodularity as presented in Definition 5 was defined based on the diminishing returns property, i.e., Eq. (1). That is, the approximately submodular degree depends on how large a deviation of ϵ the diminishing returns property can hold with.

Definition 5 (ϵ-Diminishing Returns [26])
Let $\epsilon \geq 0$. A set function $f : 2^V \to \mathbb{R}$ satisfies the ϵ-diminishing returns property, if for any $X \subseteq Y \subseteq V$ and $v \notin Y$,

$$f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y) - \epsilon. \tag{3}$$

A set function f satisfies the ϵ-diminishing returns property implies that adding an element to a set Y helps at most ϵ more than adding it to a subset X of Y. It is easy to see that f is submodular iff $\epsilon = 0$. The standard greedy algorithm has been proved to find a subset X with $f(X) \geq (1 - 1/e) \cdot (\mathrm{OPT} - k\epsilon)$ [26].

In [8], the submodularity ratio as presented in Definition 6 was introduced to measure the closeness of a set function f to submodularity.

Definition 6 (Submodularity Ratio [8])
Let $f : 2^V \to \mathbb{R}$ be a set function. The submodularity ratio of f with respect to a set $X \subseteq V$ and a parameter $l \geq 1$ is

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

Intuitively, the submodularity ratio captures how much more f can increase by adding any set S with at most l elements to any subset L of X, compared with the combined increment on f by adding the individual elements of S to L. It is easy to see from Eq. (2) that f is submodular iff $\gamma_{X,l}(f) = 1$ for any X and l. When the meaning of f is clear in the paper, we will omit f and use $\gamma_{X,l}$ for short. The standard greedy algorithm has been proved to find a subset X with $f(X) \geq (1 - e^{-\gamma_{X,k}}) \cdot \mathrm{OPT}$ [8].

The above two notions of approximate submodularity are based on the equivalent statements, i.e., Eqs. (1) and (2), of submodularity, while in [21], the approximate submodularity of a set function f as presented in Definition 7 was defined based on the closeness to other submodular functions.

Definition 7 (ϵ-Approximate Submodularity [21])
Let $\epsilon \geq 0$. A set function $f : 2^V \to \mathbb{R}$ is ϵ-approximately submodular if there exists a submodular set function g such that $\forall X \subseteq V$,

$$(1-\epsilon) \cdot g(X) \leq f(X) \leq (1+\epsilon) \cdot g(X).$$

It is easy to see that f is submodular iff $\epsilon = 0$. The standard greedy algorithm has been proved to find a subset X with $f(X) \geq \frac{1}{1+\frac{4k\epsilon}{(1-\epsilon)^2}}(1 - e^{-1}(\frac{1-\epsilon}{1+\epsilon})^{2k}) \cdot \mathrm{OPT}$ [21].

Note that a set function satisfying Eq. (3) was also originally said to be ϵ-approximately submodular [26]. For a clearer presentation, we have renamed ϵ-approximate submodularity to ϵ-diminishing returns in Definition 5.

Next, we introduce five applications, i.e., sparse regression, sparse support selection, dictionary selection, Bayesian experimental design, and determinantal function maximization, that will be examined in this paper.

2.2.1. Sparse Regression

Sparse regression is to find a sparse approximation solution to the linear regression problem, where the solution vector can have only a few non-zero elements.

Definition 8 (Sparse Regression [8])
Given all observation variables $V = \{v_1, v_2, \ldots, v_n\}$, a predictor variable z and a budget k, to find a set of at most k observation variables maximizing the squared multiple correlation *[23], i.e.,*

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

where $\mathrm{Var}(z)$ denotes the variance of z and $\mathrm{MSE}_{z,X} = \min_{\boldsymbol{\alpha} \in \mathbb{R}^{|X|}} \mathbb{E}[(z - \sum_{i \in X} \alpha_i v_i)^2]$ denotes the mean squared error.

Note that in the definition of $\mathrm{MSE}_{z,X}$, X and its index set $\{i \mid v_i \in X\}$ are not distinguished for notational convenience. The objective function $R^2_{z,X}$, capturing the portion of the variance of z explained by variables in X, is monotone but not necessarily submodular. Let $\mathbf{C}$ be the covariance matrix between all observation variables, and $\lambda_{\min}(\mathbf{C}, m)$ be the smallest m-sparse eigenvalue of $\mathbf{C}$, i.e., the minimum eigenvalue of any $m \times m$ submatrix of $\mathbf{C}$. It has been proved [8] that the submodularity ratio of $R^2_{z,X}$ can be lower bounded as $\gamma_{X,l} \geq \lambda_{\min}(\mathbf{C}, |X| + l) \geq \lambda_{\min}(\mathbf{C}, n)$.

2.2.2. Sparse Support Selection

Sparse support selection is a general sparsity constraint problem. The goal is to maximize general concave functions under sparsity constraints.

Definition 9 (Sparse Support Selection [11])
Given a ground set $V = \{v_1, v_2, \ldots, v_n\}$, a concave function $g : \mathbb{R}^n \to \mathbb{R}$ and a budget k, to find a subset $X \subseteq V$ such that

$$\arg\max_{X \subseteq V} \left(f(X) = \max_{\mathrm{supp}(\boldsymbol{s}) \subseteq X} g(\boldsymbol{s}) - g(\mathbf{0}) \right) \quad \textit{s.t.} \quad |X| \leq k,$$

where $\mathrm{supp}(\boldsymbol{s}) = \{v_i \mid s_i \neq 0\}$ *denotes the support of* $\boldsymbol{s} \in \mathbb{R}^n$.

It is clear that sparse regression in Definition 8 is a special case. More examples include low rank optimization [24], etc. Note that $g(\mathbf{0})$ is subtracted for normalization. The objective f is monotone but not necessarily submodular. It has been proved [11] that when the concave function g is m-strongly concave on all $(|X| + l)$-sparse vectors and M-smooth on all $(|X| + 1)$-sparse vectors, the submodularity ratio of f satisfies $\gamma_{X,l} \geq m/M$.

2.2.3. Dictionary Selection

Dictionary selection generalizes sparse regression to estimate multiple predictor variables.

Definition 10 (Dictionary Selection [26])
Given all observation variables $V = \{v_1, v_2, \ldots, v_n\}$, multiple predictor variables $\{z_1, z_2, \ldots, z_m\}$ and two positive integers k and d, to find a set of at most k observation variables maximizing the average squared multiple correlation, *i.e.,*

$$\arg\max_{X \subseteq V} \left(f(X) = \frac{1}{m} \sum\nolimits_{i=1}^{m} \max_{S \subseteq X, |S| \leq d} R^2_{z_i, S} \right) \quad \textit{s.t.} \quad |X| \leq k.$$

The objective f is monotone. It has been proved [26] that f satisfies the ϵ-diminishing returns property with $\epsilon \leq 4d\mu$, where μ denotes the coherence of V, i.e., the maximum absolute correlation between any pair of observation variables.

2.2.4. Bayesian Experimental Design

In Bayesian experimental design, the goal is to select observations to maximize the quality of parameter estimation. Krause *et al.* [28] considered the Bayesian A-optimality objective function, in order to maximally reduce the variance of the posterior distribution over parameters in linear models. For a matrix $\mathbf{V} \in \mathbb{R}^{d \times n}$, let $\mathbf{V}_X \in \mathbb{R}^{d \times |X|}$ denote the submatrix of $\mathbf{V}$ with its columns indexed by $X \subseteq \{1, 2, \ldots, n\}$.

Definition 11 (Bayesian Experimental Design [28])
Given an observation matrix $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{d\times n}$*, a linear model* $\boldsymbol{y}_X = \mathbf{V}_X^{\mathrm{T}}\boldsymbol{\theta} + \boldsymbol{w}$ *and a budget* k*, where* $\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{\Lambda}^{-1})$*,* $\boldsymbol{\Lambda} = \beta^2\mathbf{I}_d$*, the Gaussian noise* $\boldsymbol{w} \sim \mathcal{N}(0, \sigma^2\mathbf{I}_{|X|})$*, and* $\mathbf{I}_j$ *denotes the identity matrix of size* j*, to find a submatrix* $\mathbf{V}_X$ *of at most* k *columns maximizing the Bayesian A-optimality objective function, i.e.,*

$$\arg\max_{X\subseteq\{1,2,\ldots,n\}} \left(f(X) = \mathrm{tr}(\boldsymbol{\Lambda}^{-1}) - \mathrm{tr}((\boldsymbol{\Lambda} + \sigma^{-2}\mathbf{V}_X\mathbf{V}_X^{\mathrm{T}})^{-1})\right) \quad \textit{s.t.} \quad |X| \le k,$$

where $\mathrm{tr}(\cdot)$ *denotes the trace of a matrix.*

Note that each $\boldsymbol{v}_i \in \mathbb{R}^d$ has been assumed to be normalized, i.e., $\|\boldsymbol{v}_i\| = 1$. The objective f is monotone, and the submodularity ratio satisfies $\gamma_{X,l} \ge \beta^2/(\|\mathbf{V}\|^2(\beta^2 + \sigma^{-2}\|\mathbf{V}\|^2))$ [4], where $\|\cdot\|$ denotes the spectral norm of a matrix.

2.2.5. Determinantal Function Maximization

In non-parametric learning, e.g., sparse Gaussian processes, the goal is to select a set of representative data points. Let $\mathbf{C} \in \mathbb{R}^{n\times n}$ be the covariance matrix parameterized by a positive definite kernel. Let $\mathbf{C}^X \in \mathbb{R}^{|X|\times|X|}$ denote the submatrix of $\mathbf{C}$ with its rows and columns indexed by $X \subseteq \{1, 2, \ldots, n\}$. The determinantal function, $f(X) = \det(\mathbf{I}_{|X|} + \sigma^{-2}\mathbf{C}^X)$, is often involved in the objective functions of non-parametric learning, e.g., [29, 30]. Bian *et al.* [4] considered the problem of maximizing the determinantal function with a size constraint.

Definition 12 (Determinantal Function Maximization [4])
Given a data matrix $\mathbf{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{d\times n}$ *with the covariance matrix* $\mathbf{C} \in \mathbb{R}^{n\times n}$*, and a budget* k*, to find a submatrix* $\mathbf{V}_X$ *of at most* k *columns maximizing the determinantal function, i.e.,*

$$\arg\max_{X\subseteq\{1,2,\ldots,n\}} \left(f(X) = \det(\mathbf{I}_{|X|} + \sigma^{-2}\mathbf{C}^X)\right) \quad \textit{s.t.} \quad |X| \le k,$$

where $\sigma > 0$*.*

Though the logarithm of f is monotone and submodular [27], the determinantal function f itself is not submodular. Let $\mathbf{A}$ denote $\mathbf{I}_n + \sigma^{-2}\mathbf{C}$. It has been proved [43] that $\gamma_{X,l} \ge (\lambda_n(\mathbf{A}) - 1)/((\lambda_1(\mathbf{A}) - 1)\prod_{i=1}^{n-1}\lambda_i(\mathbf{A}))$, where $\lambda_i(\cdot)$ denotes the i-th largest eigenvalue of a square matrix.

3. Multi-objective Evolutionary Algorithms

To examine the performance of EAs optimizing the problem classes in Definitions 1, 2 and 4, we consider a simple multi-objective EA called GSEMO-C, which is slightly modified from the algorithm GSEMO widely used in previous theoretical analyses [5, 14, 39, 40]. The GSEMO generates a

new solution (i.e., set) by bit-wise mutation in each iteration, whereas the GSEMO-C generates this new set as well as its complement in each iteration. Note that the letter "C" in GSEMO-C denotes "complement".

The GSEMO-C as presented in Algorithm 1 is used for maximizing multi-objective pseudo-Boolean problems with m objective functions $f_i : \{0,1\}^n \to \mathbb{R}$ ($1 \leq i \leq m$). Note that a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ naturally characterizes a set function $f : 2^V \to \mathbb{R}$, since a subset X of V can be naturally represented by a Boolean vector $\boldsymbol{x} \in \{0,1\}^n$, where the i-th bit $x_i = 1$ means that $v_i \in X$, and $x_i = 0$ means that $v_i \notin X$. Throughout the paper, we will not distinguish $\boldsymbol{x} \in \{0,1\}^n$ and its corresponding subset for notational convenience.

Before introducing the GSEMO-C, we first introduce some basic concepts in multi-objective maximization. Since the objectives to be maximized are usually conflicted, there is no canonical complete order on the solution space. The comparison between two solutions relies on the *domination* relationship. For two solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\boldsymbol{x}$ *weakly dominates* $\boldsymbol{x}'$ (i.e., $\boldsymbol{x}$ is *better* than $\boldsymbol{x}'$, denoted by $\boldsymbol{x} \succeq \boldsymbol{x}'$) if $\forall 1 \leq i \leq m$, $f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}')$; $\boldsymbol{x}$ *dominates* $\boldsymbol{x}'$ (i.e., $\boldsymbol{x}$ is *strictly better* than $\boldsymbol{x}'$, denoted by $\boldsymbol{x} \succ \boldsymbol{x}'$) if $\boldsymbol{x} \succeq \boldsymbol{x}'$ and $f_i(\boldsymbol{x}) > f_i(\boldsymbol{x}')$ for some i. But if neither $\boldsymbol{x}$ is better than $\boldsymbol{x}'$ nor $\boldsymbol{x}'$ is better than $\boldsymbol{x}$, we say that they are *incomparable*. A solution is *Pareto optimal* if no other solution dominates it. The set of objective vectors of all the Pareto optimal solutions constitutes the *Pareto front*. The goal of multi-objective optimization is to find the Pareto front, that is, to find at least one corresponding solution for each objective vector in the Pareto front.

The procedure of the GSEMO-C is presented in Algorithm 1. Starting from a random solution (lines 1-2), it iteratively tries to improve the quality of the solutions in the population P (lines 3-12). In each iteration, a new solution $\boldsymbol{x}'$ is generated by randomly flipping bits of an archived solution $\boldsymbol{x}$ selected from the current population P (lines 4-5); the complementary set $\boldsymbol{x}'' = V \setminus \boldsymbol{x}'$ of $\boldsymbol{x}'$ is also generated (line 6); these two newly generated solutions are then used to update the population P (lines 7-11). In the updating procedure, if $\boldsymbol{y} \in \{\boldsymbol{x}', \boldsymbol{x}''\}$ is not dominated by (i.e., not strictly worse than) any previously archived solution (line 8), it will be added into P, and meanwhile those previously archived solutions weakly dominated by (i.e., worse than) $\boldsymbol{y}$ will be removed from P (line 9). It is easy to see that the population P will always contain a set of incomparable solutions due to the domination-based comparison.

Compared with the GSEMO [5, 14, 39, 40], the GSEMO-C additionally performs line 6, i.e., generates the complement $\boldsymbol{x}''$ of the new solution $\boldsymbol{x}'$. Also, both $\boldsymbol{x}'$ and $\boldsymbol{x}''$, rather than only $\boldsymbol{x}'$, are used to update the population.

For optimizing the problems in Definitions 1, 2 and 4 by the GSEMO-C, each problem is transformed

Algorithm 1 GSEMO-C

Given m pseudo-Boolean objective functions $f_1, f_2, \ldots, f_m$, where $f_i : \{0,1\}^n \to \mathbb{R}$, the GSEMO-C consists of the following steps:

Choose $\boldsymbol{x} \in \{0,1\}^n$ uniformly at random;
$P \leftarrow \{\boldsymbol{x}\}$;
repeat
  Choose $\boldsymbol{x}$ from P uniformly at random;
  Create $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ with probability $1/n$;
  Create $\boldsymbol{x}'' \leftarrow V \setminus \boldsymbol{x}'$;
  for $\boldsymbol{y} \in \{\boldsymbol{x}', \boldsymbol{x}''\}$
    if $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succ \boldsymbol{y}$ **then**
      $P \leftarrow (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{y} \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{y}\}$
    end if
  end for
until some criterion is met

into a bi-objective maximization problem

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

where $f_1(\boldsymbol{x}) = f(\boldsymbol{x})$ and $f_2(\boldsymbol{x}) = -|\boldsymbol{x}|$. That is, the GSEMO-C is to maximize the objective function f and minimize the subset size $|\boldsymbol{x}|$ simultaneously. Note that $|\boldsymbol{x}| = \sum_{i=1}^{n} x_i$ denotes the number of 1-bits of a solution $\boldsymbol{x}$. When the GSEMO-C terminates after running a number of iterations, the best solution w.r.t. the original single-objective problem in the resulting population P will be returned. For the problem in Definition 1, the solution with the largest f value in P (i.e., $\arg\max_{\boldsymbol{x} \in P} f(\boldsymbol{x})$) will be returned. For the problem in Definitions 2 and 4, the solution with the largest f value satisfying the size constraint in P (i.e., $\arg\max_{\boldsymbol{x} \in P, |\boldsymbol{x}| \leq k} f(\boldsymbol{x})$) will be returned. The running time of the GSEMO-C is measured by the number of fitness evaluations until the best solution w.r.t. the original single-objective problem in the population reaches some approximation guarantee for the first time. Since only the new solutions $\boldsymbol{x}'$ and $\boldsymbol{x}''$ need to be evaluated in each iteration of the GSEMO-C, the number of fitness evaluations is just the double of the number of iterations of the GSEMO-C.

Note that multi-objective optimization here is just an intermediate process, which has been shown helpful for solving some single-objective combinatorial optimization problems [14, 36, 39, 41]. We still focus on the quality of the best solution w.r.t. the original single-objective problem, in the population found by the GSEMO-C, rather than the quality of the population w.r.t. the transformed bi-objective optimization problem.

4. Analysis on Submodular Function Maximization

In this section, we theoretically analyze the performance of the GSEMO-C for maximizing submodular, but not necessarily monotone, functions.

4.1. Without Constraints

First, we consider the problem class in Definition 1, i.e., maximizing non-monotone submodular functions without constraints. We prove in Theorem 1 that the GSEMO-C can achieve a constant approximation ratio of nearly $1/3$ in polynomial expected time.

Theorem 1
For maximizing a non-monotone submodular function without constraints, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $f(\boldsymbol{x}) \geq (\frac{1}{3} - \frac{\epsilon}{n}) \cdot \mathrm{OPT}$ is $O(\frac{1}{\epsilon} n^4 \log n)$, where $\epsilon > 0$.

The proof relies on Lemma 1, which shows that it is always possible to improve a solution by inserting or deleting one element until a good approximation has been achieved. This lemma is extracted from Lemma 3.4 in [12].

Lemma 1 ([12])
Let $\boldsymbol{x} \in \{0,1\}^n$ be a solution such that no solution $\boldsymbol{x}'$ with the objective value $f(\boldsymbol{x}') > (1 + \frac{\epsilon}{n^2}) \cdot f(\boldsymbol{x})$ can be achieved by inserting one element into $\boldsymbol{x}$ or deleting one element from $\boldsymbol{x}$, where $\epsilon > 0$. Then $\max\{f(\boldsymbol{x}), f(V \setminus \boldsymbol{x})\} \geq (\frac{1}{3} - \frac{\epsilon}{n}) \cdot \mathrm{OPT}$.

Inspired from the proof of Theorem 4 in [13], the intuition of our proof is to follow the behavior of the local search algorithm [12], which iteratively tries to improve a solution by inserting or deleting one element.

Proof of Theorem 1. We divide the optimization process into three phases: (1) starts from an initial random solution and finishes after finding the all-0s solution $\mathbf{0}$ (i.e., $\emptyset$); (2) starts after phase (1) and finishes after finding a solution with the objective value at least OPT/n; (3) starts after phase (2) and finishes after finding a solution with the desired approximation guarantee. We analyze the expected running time of each phase, respectively, and then sum up them to get an upper bound on the total expected running time of the GSEMO-C.

For phase (1), we consider the minimum number of 1-bits of the solutions in the population P, denoted by $J_{\min}$. That is, $J_{\min} = \min\{|\boldsymbol{x}| \mid \boldsymbol{x} \in P\}$. Assume that currently $J_{\min} = i > 0$, and let $\boldsymbol{x}$ be the corresponding solution, i.e., $|\boldsymbol{x}| = i$. It is easy to see that $J_{\min}$ cannot increase because $\boldsymbol{x}$ cannot

be weakly dominated by a solution with more 1-bits. In each iteration of the GSEMO-C, to decrease $J_{\min}$, it is sufficient to select $\boldsymbol{x}$ in line 4 of Algorithm 1 and flip only one 1-bit of $\boldsymbol{x}$ in line 5. This is because the newly generated solution $\boldsymbol{x}'$ now has the smallest number of 1-bits (i.e., $|\boldsymbol{x}'| = i-1$) and no solution in P can dominate it; thus it will be included into P. Let $P_{\max}$ denote the largest size of P during the run of the GSEMO-C. The probability of selecting $\boldsymbol{x}$ in line 4 of Algorithm 1 is $\frac{1}{|P|} \geq \frac{1}{P_{\max}}$ due to uniform selection, and the probability of flipping only one 1-bit of $\boldsymbol{x}$ in line 5 is $\frac{i}{n}(1-\frac{1}{n})^{n-1} \geq \frac{i}{en}$, since $\boldsymbol{x}$ has i 1-bits. Thus, the probability of decreasing $J_{\min}$ by at least 1 in each iteration of the GSEMO-C is at least $\frac{i}{enP_{\max}}$. Note that $J_{\min} \leq n$. We can then get that the expected number of iterations of phase (1) (i.e., $J_{\min}$ reaches 0) is at most

$$\sum_{i=1}^{n} \frac{enP_{\max}}{i} = O(nP_{\max}\log n).$$

Note that the solution $\mathbf{0}$ will always be kept in P once generated, since it has the smallest subset size 0 and no other solution can weakly dominate it.

For phase (2), it is sufficient that in one iteration of the GSEMO-C, the solution $\mathbf{0}$ is selected in line 4, and only a specific 0-bit corresponding to the best single element v^* (i.e., $v^* \in \arg\max_{v\in V} f(\{v\})$) is flipped in line 5. That is, the solution $\{v^*\}$ is generated. Since the objective function f is submodular and non-negative, we easily have $f(\{v^*\}) \geq \mathrm{OPT}/n$. After generating the solution $\{v^*\}$, it will be used to update the population P, which makes P always contain a solution $\boldsymbol{z}$ weakly dominating $\{v^*\}$, i.e., $f(\boldsymbol{z}) \geq f(\{v^*\}) \geq \mathrm{OPT}/n$ and $|\boldsymbol{z}| \leq |\{v^*\}| = 1$. Thus, we only need to analyze the expected number of iterations of the GSEMO-C until generating the solution $\{v^*\}$. Since the probability of selecting $\mathbf{0}$ in line 4 of the GSEMO-C is at least $\frac{1}{P_{\max}}$ and the probability of flipping only a specific 0-bit in line 5 is $\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$, the expected number of iterations of phase (2) is $O(nP_{\max})$.

As in [12], we call a solution $\boldsymbol{x}$ a $(1+\alpha)$-approximate local optimum if $f(\boldsymbol{x}\setminus\{v\}) \leq (1+\alpha)\cdot f(\boldsymbol{x})$ for any $v \in \boldsymbol{x}$ and $f(\boldsymbol{x}\cup\{v\}) \leq (1+\alpha)\cdot f(\boldsymbol{x})$ for any $v \notin \boldsymbol{x}$. By Lemma 1, we know that a $(1+\frac{\epsilon}{n^2})$-approximate local optimum $\boldsymbol{x}$ satisfies $\max\{f(\boldsymbol{x}), f(V\setminus\boldsymbol{x})\} \geq (\frac{1}{3}-\frac{\epsilon}{n})\cdot\mathrm{OPT}$. For phase (3), we thus only need to analyze the expected number of iterations until generating a $(1+\frac{\epsilon}{n^2})$-approximate local optimum $\boldsymbol{x}'$. This is because both $\boldsymbol{x}'$ and $V\setminus\boldsymbol{x}'$ will be used to update the population P, and then for either one of $\boldsymbol{x}'$ and $V\setminus\boldsymbol{x}'$, P will always contain one solution weakly dominating it, which implies that $\max\{f(\boldsymbol{x}) \mid \boldsymbol{x}\in P\} \geq \max\{f(\boldsymbol{x}'), f(V\setminus\boldsymbol{x}')\} \geq (\frac{1}{3}-\frac{\epsilon}{n})\cdot\mathrm{OPT}$. We then consider the largest f value of the solutions in the population P, denoted by $J_{\max}$. That is, $J_{\max} = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x}\in P\}$. After phase (2), $J_{\max} \geq \mathrm{OPT}/n$, and let $\boldsymbol{x}$ be the corresponding solution, i.e., $f(\boldsymbol{x}) = J_{\max}$. It is obvious that $J_{\max}$ cannot decrease, because $\boldsymbol{x}$ cannot be weakly dominated by a solution with a smaller f value. As long as $\boldsymbol{x}$ is not a $(1+\frac{\epsilon}{n^2})$-approximate local optimum, we know that a new solution $\boldsymbol{x}'$ with $f(\boldsymbol{x}') > (1+\frac{\epsilon}{n^2})f(\boldsymbol{x}) = (1+\frac{\epsilon}{n^2})J_{\max}$ can be generated through selecting $\boldsymbol{x}$ in line 4 of Algorithm 1 and flipping only one specific 1-bit (i.e., deleting one specific element from $\boldsymbol{x}$) or one

specific 0-bit (i.e., adding one specific element into $\boldsymbol{x}$) in line 5, the probability of which is at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{enP_{\max}}$. Since $\boldsymbol{x}'$ now has the largest f value and no other solution in P can dominate it, it will be included into P. Thus, $J_{\max}$ can increase by at least a factor of $(1+\frac{\epsilon}{n^2})$ with probability at least $\frac{1}{enP_{\max}}$ in each iteration. Such an increase on $J_{\max}$ is called a successful step. Thus, a successful step needs at most $enP_{\max}$ expected number of iterations. It is also easy to see that until generating a $(1+\frac{\epsilon}{n^2})$-approximate local optimum, the number of successful steps is at most $\log_{1+\frac{\epsilon}{n^2}} \frac{\mathrm{OPT}}{\mathrm{OPT}/n} = O(\frac{1}{\epsilon}n^2 \log n)$. Thus, the expected number of iterations of phase (3) is at most

$$enP_{\max} \cdot O\left(\frac{1}{\epsilon}n^2 \log n\right) = O\left(\frac{1}{\epsilon}n^3 P_{\max} \log n\right).$$

From the procedure of the GSEMO-C, we know that the solutions maintained in P must be incomparable. Thus, each value of one objective can correspond to at most one solution in P. Because the second objective $f_2(\boldsymbol{x}) = -|\boldsymbol{x}|$ can only belong to $\{0,-1,\ldots,-n\}$, we have $P_{\max} \leq n+1$. Hence, the expected running time of the GSEMO-C for finding a solution with the objective function value at least $(\frac{1}{3}-\frac{\epsilon}{n}) \cdot \mathrm{OPT}$ is

$$O(nP_{\max}\log n) + O(nP_{\max}) + O\left(\frac{1}{\epsilon}n^3 P_{\max}\log n\right) = O\left(\frac{1}{\epsilon}n^4 \log n\right).$$

□

Note that in parallel with our work, Friedrich *et al.* [15] analyzed the performance of the (1+1)-EA using a new and novel mutation operator for solving this problem class. The mutation operator is a heavy-tailed mutation operator, which samples $m \in \{1,2,\ldots,n\}$ according to a power-law distribution, and then flips m bits of a solution chosen uniformly at random. The power-law distribution has a parameter $\beta > 1$, that can be chosen arbitrarily close to 1. They proved that the (1+1)-EA can achieve an approximation ratio of $(\frac{1}{3}-\frac{\epsilon}{n})$ in $O(\frac{1}{\epsilon}n^3 \log \frac{n}{\epsilon} + n^\beta)$ expected running time.

4.2. With a Size Constraint

Next, we consider the problem class in Definition 2, i.e., maximizing submodular and approximately monotone functions with a size constraint. As in previous analyses (e.g., [6, 16]), we may assume that there is a set D of k "dummy" elements whose marginal contribution to any set is 0, i.e., for any $X \subseteq V$, $f(X) = f(X \setminus D)$. Theorem 2 gives the approximation guarantee of the GSEMO-C.

Theorem 2

For maximizing a submodular function f with a size constraint k, where f is ϵ-approximately monotone as in Definition 3, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-1/e) \cdot (\mathrm{OPT} - k\epsilon)$ is $O(n^2(\log n + k))$.

The proof relies on Lemma 2, that for any $\boldsymbol{x} \in \{0,1\}^n$ with $|\boldsymbol{x}| < k$, there always exists another element, the inclusion of which can bring an improvement on f roughly proportional to the current distance to the optimum.

Lemma 2

Assume that a set function f is submodular and ϵ-approximately monotone as in Definition 3. For any $\boldsymbol{x} \in \{0,1\}^n$ with $|\boldsymbol{x}| < k$, there exists one element $v \notin \boldsymbol{x}$ such that

$$f(\boldsymbol{x} \cup \{v\}) - f(\boldsymbol{x}) \geq \frac{1}{k}(\mathrm{OPT} - f(\boldsymbol{x})) - \epsilon, \tag{4}$$

where k is the size constraint.

Proof. Let $\boldsymbol{x}^*$ be an optimal solution, i.e., $f(\boldsymbol{x}^*) = \mathrm{OPT}$. We denote the elements in $\boldsymbol{x} \setminus \boldsymbol{x}^*$ by $u_1^*, u_2^*, \ldots, u_m^*$, where $m = |\boldsymbol{x} \setminus \boldsymbol{x}^*|$. Note that $m < k$ as $|\boldsymbol{x}| < k$. Because f is ϵ-approximately monotone, we have

$$\begin{aligned} f(\boldsymbol{x}^* \cup \boldsymbol{x}) = f(\boldsymbol{x}^* \cup \{u_1^*, \ldots, u_m^*\}) &\geq f(\boldsymbol{x}^* \cup \{u_1^*, \ldots, u_{m-1}^*\}) - \epsilon \\ &\geq \cdots \geq f(\boldsymbol{x}^*) - m\epsilon \geq f(\boldsymbol{x}^*) - k\epsilon, \end{aligned} \tag{5}$$

where the first three inequalities hold by Definition 3.

We denote the elements in $\boldsymbol{x}^* \setminus \boldsymbol{x}$ by $v_1^*, v_2^*, \ldots, v_l^*$, where $l = |\boldsymbol{x}^* \setminus \boldsymbol{x}| \leq k$. Then, we have

$$\begin{aligned} f(\boldsymbol{x}^*) - f(\boldsymbol{x}) - k\epsilon &\leq f(\boldsymbol{x} \cup \boldsymbol{x}^*) - f(\boldsymbol{x}) \\ &= f(\boldsymbol{x} \cup \{v_1^*, \ldots, v_l^*\}) - f(\boldsymbol{x}) \\ &= \sum_{j=1}^{l} \left(f(\boldsymbol{x} \cup \{v_1^*, \ldots, v_j^*\}) - f(\boldsymbol{x} \cup \{v_1^*, \ldots, v_{j-1}^*\}) \right) \\ &\leq \sum_{j=1}^{l} \left(f(\boldsymbol{x} \cup \{v_j^*\}) - f(\boldsymbol{x}) \right), \end{aligned} \tag{6}$$

where the first inequality holds by Eq. (5), the first equality holds by the definition of $\boldsymbol{x}^* \setminus \boldsymbol{x}$, and the last inequality holds by Eq. (1) since f is submodular. Let $v^* = \arg\max_{v \in V \setminus \boldsymbol{x}} f(\boldsymbol{x} \cup \{v\})$. Eq. (6) implies that

$$f(\boldsymbol{x}^*) - f(\boldsymbol{x}) - k\epsilon \leq l \left(f(\boldsymbol{x} \cup \{v^*\}) - f(\boldsymbol{x}) \right).$$

Due to the existence of k dummy elements and $|\boldsymbol{x}| < k$, there must exist one dummy element $v \notin \boldsymbol{x}$ satisfying $f(\boldsymbol{x} \cup \{v\}) - f(\boldsymbol{x}) = 0$; this implies that $f(\boldsymbol{x} \cup \{v^*\}) - f(\boldsymbol{x}) \geq 0$. As $l \leq k$, we have

$$f(\boldsymbol{x}^*) - f(\boldsymbol{x}) - k\epsilon \leq k \left(f(\boldsymbol{x} \cup \{v^*\}) - f(\boldsymbol{x}) \right),$$

leading to

$$f(\boldsymbol{x} \cup \{v^*\}) - f(\boldsymbol{x}) \geq \frac{1}{k}(\mathrm{OPT} - f(\boldsymbol{x})) - \epsilon.$$

□

Inspired from the proof of Theorem 2 in [13], our proof idea is to follow the behavior of the standard greedy algorithm, which iteratively adds one element with the currently largest improvement on f.

Proof of Theorem 2. We divide the optimization process into two phases: (1) starts from an initial random solution and finishes after finding the special solution $\mathbf{0}$; (2) starts after phase (1) and finishes after finding a solution with the desired approximation guarantee. As the analysis of phase (1) in the proof of Theorem 1, we know that the population P will contain the solution $\mathbf{0}$ after $O(nP_{\max}\log n)$ iterations in expectation.

For phase (2), we consider a quantity $J_{\max}$, which is defined as

$$J_{\max} = \max\left\{ j \in \{0,1,\ldots,k\} \mid \exists \boldsymbol{x} \in P : |\boldsymbol{x}| \leq j \wedge f(\boldsymbol{x}) \geq \left(1 - \left(1 - \frac{1}{k}\right)^j\right) \cdot (\mathrm{OPT} - k\epsilon) \right\}.$$

That is, $J_{\max}$ denotes the maximum value of $j \in \{0,1,\ldots,k\}$ such that in the population P, there exists a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq j$ and $f(\boldsymbol{x}) \geq (1-(1-\frac{1}{k})^j) \cdot (\mathrm{OPT} - k\epsilon)$. We analyze the expected number of iterations until $J_{\max} = k$, which implies that there exists one solution $\boldsymbol{x}$ in P satisfying that $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-(1-\frac{1}{k})^k) \cdot (\mathrm{OPT} - k\epsilon) \geq (1-1/e) \cdot (\mathrm{OPT} - k\epsilon)$. That is, the desired approximation guarantee is reached.

The current value of $J_{\max}$ is at least 0, since the population P contains the solution $\mathbf{0}$, which will always be kept in P once generated. Assume that currently $J_{\max} = i < k$. Let $\boldsymbol{x}$ be a corresponding solution with the value i, i.e., $|\boldsymbol{x}| \leq i$ and $f(\boldsymbol{x}) \geq (1-(1-\frac{1}{k})^i) \cdot (\mathrm{OPT} - k\epsilon)$. It is easy to see that $J_{\max}$ cannot decrease because cleaning $\boldsymbol{x}$ from P (line 9 of Algorithm 1) implies that $\boldsymbol{x}$ is weakly dominated by a newly generated solution $\boldsymbol{y}$, which must satisfy that $|\boldsymbol{y}| \leq |\boldsymbol{x}|$ and $f(\boldsymbol{y}) \geq f(\boldsymbol{x})$. By Lemma 2, we know that flipping one specific 0-bit of $\boldsymbol{x}$ (i.e., adding a specific element) can generate a new solution $\boldsymbol{x}'$, which satisfies $f(\boldsymbol{x}') - f(\boldsymbol{x}) \geq \frac{1}{k}(\mathrm{OPT} - f(\boldsymbol{x})) - \epsilon$. Then, we have

$$f(\boldsymbol{x}') \geq \left(1 - \frac{1}{k}\right) f(\boldsymbol{x}) + \frac{1}{k} \cdot \mathrm{OPT} - \epsilon \geq \left(1 - \left(1 - \frac{1}{k}\right)^{i+1}\right) \cdot (\mathrm{OPT} - k\epsilon),$$

where the last inequality is derived by $f(\boldsymbol{x}) \geq (1-(1-\frac{1}{k})^i) \cdot (\mathrm{OPT} - k\epsilon)$. Since $|\boldsymbol{x}'| = |\boldsymbol{x}| + 1 \leq i+1$, $\boldsymbol{x}'$ will be included into P; otherwise, $\boldsymbol{x}'$ must be dominated by one solution in P (line 8 of Algorithm 1), and this implies that $J_{\max}$ has already been larger than i, contradicting with the assumption $J_{\max} = i$. After including $\boldsymbol{x}'$, $J_{\max} \geq i+1$. Thus, $J_{\max}$ can increase by at least 1 in one iteration with probability at least $\frac{1}{P_{\max}} \cdot \frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{enP_{\max}}$, where $\frac{1}{P_{\max}}$ is a lower bound on the probability of selecting $\boldsymbol{x}$ in line 4 of Algorithm 1 and $\frac{1}{n}(1-\frac{1}{n})^{n-1}$ is the probability of flipping a specific bit of $\boldsymbol{x}$ while keeping other bits unchanged in line 5. This implies that it needs at most $enP_{\max}$ expected number of iterations to increase $J_{\max}$. Thus, after at most $k \cdot enP_{\max}$ iterations in expectation, $J_{\max}$ must have reached k.

As the proof of Theorem 1, we know that $P_{\max} \leq n+1$. Thus, by summing up the expected running time of two phases, we get that the expected running time of the GSEMO-C for finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-1/e)\cdot(\mathrm{OPT}-k\epsilon)$ is $O(nP_{\max}\log n + knP_{\max}) = O(n^2(\log n + k))$. □

Note that the approximation guarantee, i.e., $f(\boldsymbol{x}) \geq (1-1/e)\cdot(\mathrm{OPT}-k\epsilon)$, by the GSEMO-C reaches the best known one, which was previously obtained by the standard greedy algorithm [28]. Particularly, when the objective function is monotone, the parameter ϵ in Definition 3 equals 0, and thus the approximation ratio becomes $1-1/e$, which is optimal in general [33], and also consistent with the previous result in [13]. For the application of sensor placement with the mutual information as the objective function, which is submodular but not necessarily monotone, the GSEMO-C has a bounded approximation guarantee, because the mutual information can be guaranteed to be ϵ-approximately monotone [28], where ϵ depends on the discretization level of locations.

5. Analysis on Monotone Approximately Submodular Function Maximization

In this section, we analyze the performance of the GSEMO-C for maximizing monotone and approximately submodular functions with a size constraint, which has various applications as introduced in Section 2.2. We prove the polynomial-time approximation guarantee of the GSEMO-C w.r.t. each notion of approximate submodularity in Definitions 5-7, respectively.

5.1. ϵ-Diminishing Returns

First, we consider the case that the objective function satisfies the ϵ-diminishing returns property in Definition 5. Theorem 3 gives the approximation guarantee of the GSEMO-C.

Theorem 3

For maximizing a monotone function f with a size constraint k, where f satisfies the ϵ-diminishing returns property as in Definition 5, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-1/e)\cdot(\mathrm{OPT}-k\epsilon)$ is $O(n^2(\log n + k))$.

The proof relies on Lemma 3, which states that any $\boldsymbol{x} \in \{0,1\}^n$ can be improved by at least roughly $(\mathrm{OPT} - f(\boldsymbol{x}))/k$ through adding a specific element. The proof of Lemma 3 is similar to that of Lemma 2. The main difference is that the two inequalities in Eq. (6) utilize the ϵ-approximately monotone and the diminishing returns properties, respectively, whereas that in Eq. (8) utilize the monotone and the ϵ-diminishing returns properties, respectively.

Lemma 3

Assume that a set function f is monotone and satisfies the ϵ-diminishing returns property as in Definition 5. For any $\boldsymbol{x} \in \{0,1\}^n$, there exists one element $v \notin \boldsymbol{x}$ such that

$$f(\boldsymbol{x} \cup \{v\}) - f(\boldsymbol{x}) \geq \frac{1}{k}(\mathrm{OPT} - f(\boldsymbol{x})) - \epsilon, \tag{7}$$

where k is the size constraint.

Proof. Let $\boldsymbol{x}^*$ be an optimal solution, i.e., $f(\boldsymbol{x}^*) = \mathrm{OPT}$. We denote the elements in $\boldsymbol{x}^* \setminus \boldsymbol{x}$ by $v_1^*, v_2^*, \ldots, v_l^*$, where $|\boldsymbol{x}^* \setminus \boldsymbol{x}| = l \leq k$. Then, we have

$$\begin{aligned} f(\boldsymbol{x}^*) - f(\boldsymbol{x}) &\leq f(\boldsymbol{x} \cup \boldsymbol{x}^*) - f(\boldsymbol{x}) \\ &= f(\boldsymbol{x} \cup \{v_1^*, \ldots, v_l^*\}) - f(\boldsymbol{x}) \\ &= \sum_{j=1}^{l} \left(f(\boldsymbol{x} \cup \{v_1^*, \ldots, v_j^*\}) - f(\boldsymbol{x} \cup \{v_1^*, \ldots, v_{j-1}^*\}) \right) \\ &\leq \sum_{j=1}^{l} \left(f(\boldsymbol{x} \cup \{v_j^*\}) - f(\boldsymbol{x}) + \epsilon \right), \end{aligned} \tag{8}$$

where the first inequality holds by the monotonicity of f, the first equality holds by the definition of $\boldsymbol{x}^* \setminus \boldsymbol{x}$, and the last inequality is derived by Definition 5 since f satisfies the ϵ-diminishing returns property. Let $v^* = \arg\max_{v \in \boldsymbol{x}^* \setminus \boldsymbol{x}} f(\boldsymbol{x} \cup \{v\})$. Then, we have

$$f(\boldsymbol{x} \cup \{v^*\}) - f(\boldsymbol{x}) \geq \frac{1}{l}(f(\boldsymbol{x}^*) - f(\boldsymbol{x})) - \epsilon \geq \frac{1}{k}(\mathrm{OPT} - f(\boldsymbol{x})) - \epsilon.$$

□

Thus, the proof of Theorem 3 can be accomplished in the same way as that of Theorem 2. This is because the proof of Theorem 2 utilizes a quantity $J_{\max}$ based on Eq. (4), while Eq. (4) still holds here as Eq. (7) in Lemma 3.

Note that this approximation guarantee, i.e., $f(\boldsymbol{x}) \geq (1-1/e) \cdot (\mathrm{OPT} - k\epsilon)$, obtained by the GSEMO-C reaches the best known one, which was previously obtained by the standard greedy algorithm [26]. Particularly, when the objective function is submodular, the diminishing returns property holds, i.e., $\epsilon = 0$, and thus the approximation ratio reaches the optimal one, $1 - 1/e$.

For the application of dictionary selection in Definition 10 where the objective function f is monotone but not necessarily submodular, as f satisfies the ϵ-diminishing returns property with $\epsilon \leq 4d\mu$ [26], we have:

Corollary 1

For dictionary selection in Definition 10, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1 - 1/e) \cdot (\mathrm{OPT} - 4dk\mu)$ is $O(n^2(\log n + k))$, where μ denotes the coherence of V, i.e., the maximum absolute correlation between any pair of observation variables.

5.2. Submodularity Ratio

Next, we prove the approximation guarantee of the GSEMO-C w.r.t. the submodularity ratio presented in Definition 6.

Theorem 4

For maximizing a monotone function f with a size constraint k, where f is not necessarily submodular, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$, where $\gamma_{\min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$ and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k as in Definition 6.

The proof relies on Lemma 4, which shows that any $\boldsymbol{x} \in \{0,1\}^n$ can be improved by adding a specific element such that the increment on f is proportional to $(\mathrm{OPT} - f(\boldsymbol{x}))$ and depends on the submodularity ratio $\gamma_{\boldsymbol{x},k}$.

Lemma 4 ([42])

Assume that a set function f is monotone but not necessarily submodular. For any $\boldsymbol{x} \in \{0,1\}^n$, there exists one element $v \notin \boldsymbol{x}$ such that

$$f(\boldsymbol{x} \cup \{v\}) - f(\boldsymbol{x}) \geq \frac{\gamma_{\boldsymbol{x},k}}{k}(\mathrm{OPT} - f(\boldsymbol{x})), \tag{9}$$

where k is the size constraint, and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k as in Definition 6.

The proof of Theorem 4 is similar to that of Theorem 2. The main difference is that a different inductive inequality on f is used in the definition of the quantity $J_{\max}$, as Eq. (4) in Lemma 2 changes to Eq. (9) in Lemma 4. For concise illustration, we will mainly show the difference in the proof of Theorem 4.

Proof of Theorem 4. The proof is similar to that of Theorem 2. We use a different $J_{\max}$, which is defined as

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

It is easy to verify that $J_{\max} = k$ implies that the desired approximation guarantee is reached, because there must exist one solution $\boldsymbol{x}$ in P satisfying that $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-(1-\frac{\gamma_{\min}}{k})^k) \cdot \mathrm{OPT} \geq (1-e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$. Assume that currently $J_{\max} = i < k$ and $\boldsymbol{x}$ is a corresponding solution, i.e., $|\boldsymbol{x}| \leq i$ and $f(\boldsymbol{x}) \geq (1-(1-\frac{\gamma_{\min}}{k})^i) \cdot \mathrm{OPT}$. We then only need to show that flipping one specific 0-bit of $\boldsymbol{x}$ can generate a new solution $\boldsymbol{x}'$ with $f(\boldsymbol{x}') \geq (1-(1-\frac{\gamma_{\min}}{k})^{i+1}) \cdot \mathrm{OPT}$. By Lemma 4, we know that flipping one specific 0-bit of $\boldsymbol{x}$ can generate a new solution $\boldsymbol{x}'$, which satisfies $f(\boldsymbol{x}') - f(\boldsymbol{x}) \geq \frac{\gamma_{\boldsymbol{x},k}}{k}(\mathrm{OPT} - f(\boldsymbol{x}))$. Then, we have

$$f(\boldsymbol{x}') \geq \left(1 - \frac{\gamma_{\boldsymbol{x},k}}{k}\right) f(\boldsymbol{x}) + \frac{\gamma_{\boldsymbol{x},k}}{k} \cdot \mathrm{OPT}$$

$$\geq \left(1-\left(1-\frac{\gamma_{\boldsymbol{x},k}}{k}\right)\left(1-\frac{\gamma_{\min}}{k}\right)^{i}\right)\cdot \mathrm{OPT}$$
$$\geq \left(1-\left(1-\frac{\gamma_{\min}}{k}\right)^{i+1}\right)\cdot \mathrm{OPT},$$

where the second inequality holds by $f(\boldsymbol{x}) \geq (1-(1-\frac{\gamma_{\min}}{k})^i)\cdot \mathrm{OPT}$, and the last holds by $\gamma_{\boldsymbol{x},k} \geq \gamma_{\min}$, which can be derived from $|\boldsymbol{x}| < k$ and $\gamma_{\boldsymbol{x},k}$ decreasing with $\boldsymbol{x}$. Thus, the theorem holds. □

Note that it has been proved that the standard greedy algorithm can find a subset $\boldsymbol{x}$ with $|\boldsymbol{x}| = k$ and $f(\boldsymbol{x}) \geq (1-e^{-\gamma_{\boldsymbol{x},k}})\cdot \mathrm{OPT}$ [8]. Thus, Theorem 4 shows that the GSEMO-C can achieve nearly this best known approximation guarantee. Particularly, when the objective function is submodular, the submodularity ratio in Definition 6 satisfies $\forall \boldsymbol{x}, l : \gamma_{\boldsymbol{x},l} = 1$, and thus the approximation ratio, i.e., $1-e^{-\gamma_{\min}}$, by the GSEMO-C reaches the optimal one, $1-1/e$.

For the application of sparse regression in Definition 8 where the objective function $R^2_{z,\boldsymbol{x}}$ is monotone but not necessarily submodular, because the submodularity ratio of $R^2_{z,\boldsymbol{x}}$ can be lower bounded as $\gamma_{\boldsymbol{x},l} \geq \lambda_{\min}(\mathbf{C}, |\boldsymbol{x}|+l) \geq \lambda_{\min}(\mathbf{C}, n)$ [8], implying $\gamma_{\min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1}\gamma_{\boldsymbol{x},k} \geq \lambda_{\min}(\mathbf{C}, 2k-1) \geq \lambda_{\min}(\mathbf{C}, n)$, we have:

Corollary 2

For sparse regression in Definition 8, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-e^{-\lambda_{\min}(\mathbf{C},2k-1)})\cdot \mathrm{OPT} \geq (1-e^{-\lambda_{\min}(\mathbf{C},n)})\cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$, where $\lambda_{\min}(\mathbf{C}, m)$ denotes the smallest m-sparse eigenvalue of the covariance matrix $\mathbf{C}$ between all observation variables.

For the application of sparse support selection in Definition 9 where the objective function $f(\boldsymbol{x}) = \max_{\mathrm{supp}(\boldsymbol{s})\subseteq\boldsymbol{x}} g(\boldsymbol{s}) - g(\mathbf{0})$ is monotone but not necessarily submodular, the submodularity ratio of f satisfies $\gamma_{\boldsymbol{x},l} \geq m/M$, when the concave function g is m-strongly concave on all $(|\boldsymbol{x}|+l)$-sparse vectors and M-smooth on all $(|\boldsymbol{x}|+1)$-sparse vectors [11]. Thus, $\gamma_{\min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1}\gamma_{\boldsymbol{x},k} \geq m/M$, when g is m-strongly concave on all $(2k-1)$-sparse vectors and M-smooth on all k-sparse vectors. We have:

Corollary 3

For sparse support selection in Definition 9 where the concave function g is m-strongly concave on all $(2k-1)$-sparse vectors and M-smooth on all k-sparse vectors, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1-e^{-m/M})\cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$.

For the application of Bayesian experimental design in Definition 11 where the objective function f is monotone but not necessarily submodular, because the submodularity ratio of f satisfies $\forall \boldsymbol{x}, l : \gamma_{\boldsymbol{x},l} \geq \beta^2/(\|\mathbf{V}\|^2(\beta^2+\sigma^{-2}\|\mathbf{V}\|^2))$ [4], implying $\gamma_{\min} \geq \beta^2/(\|\mathbf{V}\|^2(\beta^2+\sigma^{-2}\|\mathbf{V}\|^2))$, we have:

Corollary 4

For Bayesian experimental design in Definition 11, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1 - e^{-\beta^2/(\|\mathbf{V}\|^2(\beta^2+\sigma^{-2}\|\mathbf{V}\|^2))}) \cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$.

For the application of determinantal function maximization in Definition 12 where the objective function f is monotone but not necessarily submodular, because the submodularity ratio of f satisfies $\forall \boldsymbol{x}, l : \gamma_{\boldsymbol{x},l} \geq (\lambda_n(\mathbf{A}) - 1)/((\lambda_1(\mathbf{A}) - 1)\prod_{i=1}^{n-1} \lambda_i(\mathbf{A}))$ [43], implying $\gamma_{\min} \geq (\lambda_n(\mathbf{A}) - 1)/((\lambda_1(\mathbf{A}) - 1)\prod_{i=1}^{n-1} \lambda_i(\mathbf{A}))$, we have:

Corollary 5

For determinantal function maximization in Definition 12, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1 - e^{-(\lambda_n(\mathbf{A})-1)/((\lambda_1(\mathbf{A})-1)\prod_{i=1}^{n-1} \lambda_i(\mathbf{A}))}) \cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$, where $\mathbf{A} = \mathbf{I}_n + \sigma^{-2}\mathbf{C}$ and $\lambda_i(\mathbf{A})$ denotes the i-th largest eigenvalue of $\mathbf{A}$.

5.3. ϵ-Approximate Submodularity

Finally, we consider the case that the objective function is ϵ-approximately submodular as in Definition 7. Theorem 5 gives the approximation guarantee of the GSEMO-C.

Theorem 5

For maximizing a monotone function f with a size constraint k, where f is ϵ-approximately submodular as in Definition 7, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1-(1-\frac{1}{k})^k(\frac{1-\epsilon}{1+\epsilon})^k) \cdot \mathrm{OPT} \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1-e^{-1}(\frac{1-\epsilon}{1+\epsilon})^k) \cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$.

The proof relies on the following lemma, which shows that any $\boldsymbol{x} \in \{0,1\}^n$ can be improved by adding a specific element v such that $f(\boldsymbol{x} \cup \{v\}) - \frac{1-\epsilon}{1+\epsilon} f(\boldsymbol{x})$ is proportional to the current distance to the optimum, i.e., $\mathrm{OPT} - f(\boldsymbol{x})$.

Lemma 5

Assume that a set function f is monotone and ϵ-approximately submodular as in Definition 7. For any $\boldsymbol{x} \in \{0,1\}^n$, there exists one element $v \notin \boldsymbol{x}$ such that

$$f(\boldsymbol{x} \cup \{v\}) - \frac{1-\epsilon}{1+\epsilon} f(\boldsymbol{x}) \geq \frac{1-\epsilon}{k(1+\epsilon)}(\mathrm{OPT} - f(\boldsymbol{x})), \tag{10}$$

where k is the size constraint.

Proof. Let $\boldsymbol{x}^*$ be an optimal solution, i.e., $f(\boldsymbol{x}^*) = \mathrm{OPT}$. Let $v^* = \arg\max_{v \in \boldsymbol{x}^* \setminus \boldsymbol{x}} f(\boldsymbol{x} \cup \{v\})$. As f is ϵ-approximately submodular as in Definition 7, we use g to denote one corresponding submodular

function satisfying that for all $\boldsymbol{x} \in \{0,1\}^n$, $(1-\epsilon)g(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq (1+\epsilon)g(\boldsymbol{x})$. Then, we have

$$\begin{aligned} g(\boldsymbol{x}^* \cup \boldsymbol{x}) - g(\boldsymbol{x}) &\leq \sum_{v \in \boldsymbol{x}^* \setminus \boldsymbol{x}} \big(g(\boldsymbol{x} \cup \{v\}) - g(\boldsymbol{x})\big) \\ &\leq \sum_{v \in \boldsymbol{x}^* \setminus \boldsymbol{x}} \left(\frac{1}{1-\epsilon} f(\boldsymbol{x} \cup \{v\}) - g(\boldsymbol{x})\right) \\ &\leq k \left(\frac{1}{1-\epsilon} f(\boldsymbol{x} \cup \{v^*\}) - g(\boldsymbol{x})\right), \end{aligned}$$

where the first inequality holds by the submodularity of g (i.e., Eq. (2)), the second inequality holds by $(1-\epsilon)g(\boldsymbol{x}) \leq f(\boldsymbol{x})$ for any $\boldsymbol{x}$, and the last inequality holds by the definition of v^* and $|\boldsymbol{x}^*| \leq k$. By reordering the terms, we get

$$f(\boldsymbol{x} \cup \{v^*\}) \geq \frac{1-\epsilon}{k} g(\boldsymbol{x}^* \cup \boldsymbol{x}) + \left(1 - \frac{1}{k}\right)(1-\epsilon) g(\boldsymbol{x}).$$

Because $g(\boldsymbol{x}) \geq \frac{1}{1+\epsilon} f(\boldsymbol{x})$ and $g(\boldsymbol{x}^* \cup \boldsymbol{x}) \geq \frac{1}{1+\epsilon} f(\boldsymbol{x}^* \cup \boldsymbol{x}) \geq \frac{1}{1+\epsilon} f(\boldsymbol{x}^*) = \frac{1}{1+\epsilon}\mathrm{OPT}$, where the last inequality holds by the monotonicity of f, we have

$$f(\boldsymbol{x} \cup \{v^*\}) \geq \frac{1-\epsilon}{k(1+\epsilon)}\mathrm{OPT} + \left(1 - \frac{1}{k}\right)\frac{1-\epsilon}{1+\epsilon} f(\boldsymbol{x}).$$

By reordering the terms, the lemma holds. □

The proof of Theorem 5 is also similar to that of Theorem 2, except that a different inductive inequality on f is used in the definition of the quantity $J_{\max}$, as Eq. (4) in Lemma 2 changes to Eq. (10) in Lemma 5.

Proof of Theorem 5. The proof is similar to that of Theorem 2. We use a different $J_{\max}$, which is defined as

$$J_{\max} = \max\left\{ j \in \{0,1,\ldots,k\} \mid \exists \boldsymbol{x} \in P : |\boldsymbol{x}| \leq j \wedge f(\boldsymbol{x}) \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}} \left(1 - \left(1-\frac{1}{k}\right)^j \left(\frac{1-\epsilon}{1+\epsilon}\right)^j\right) \cdot \mathrm{OPT} \right\}.$$

It is easy to verify that $J_{\max} = k$ implies that the desired approximation guarantee is reached. Assume that currently $J_{\max} = i < k$ and $\boldsymbol{x}$ is a corresponding solution, i.e., $|\boldsymbol{x}| \leq i$ and $f(\boldsymbol{x}) \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1-(1-\frac{1}{k})^i(\frac{1-\epsilon}{1+\epsilon})^i) \cdot \mathrm{OPT}$. We then only need to show that flipping one specific 0-bit of $\boldsymbol{x}$ can generate a new solution $\boldsymbol{x}'$ with $f(\boldsymbol{x}') \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1-(1-\frac{1}{k})^{i+1}(\frac{1-\epsilon}{1+\epsilon})^{i+1}) \cdot \mathrm{OPT}$. By Lemma 5, we know that flipping one specific 0-bit of $\boldsymbol{x}$ can generate a new solution $\boldsymbol{x}'$, which satisfies $f(\boldsymbol{x}') - \frac{1-\epsilon}{1+\epsilon} f(\boldsymbol{x}) \geq \frac{1-\epsilon}{k(1+\epsilon)}(\mathrm{OPT} - f(\boldsymbol{x}))$. Then, we have

$$f(\boldsymbol{x}') \geq \left(1-\frac{1}{k}\right)\frac{1-\epsilon}{1+\epsilon} f(\boldsymbol{x}) + \frac{1-\epsilon}{k(1+\epsilon)} \cdot \mathrm{OPT} \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}} \left(1 - \left(1-\frac{1}{k}\right)^{i+1} \left(\frac{1-\epsilon}{1+\epsilon}\right)^{i+1}\right) \cdot \mathrm{OPT},$$

where the second inequality is derived by applying $f(\boldsymbol{x}) \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1-(1-\frac{1}{k})^i(\frac{1-\epsilon}{1+\epsilon})^i) \cdot \mathrm{OPT}$. Thus, the theorem holds. □

Note that the standard greedy algorithm obtains the best known approximation guarantee, i.e., $f(\boldsymbol{x}) \geq \frac{1}{1+\frac{4k\epsilon}{(1-\epsilon)^2}}(1-(1-\frac{1}{k})^k(\frac{1-\epsilon}{1+\epsilon})^{2k}) \cdot \mathrm{OPT}$ [21]. Compared with this, the approximation guarantee, i.e., $f(\boldsymbol{x}) \geq \frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}(1-(1-\frac{1}{k})^k(\frac{1-\epsilon}{1+\epsilon})^{k}) \cdot \mathrm{OPT}$, of the GSEMO-C shown in Theorem 5 is slightly better, because

$$\frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}\left(1-\left(1-\frac{1}{k}\right)^k\left(\frac{1-\epsilon}{1+\epsilon}\right)^k\right) = \frac{1-\epsilon}{k(1+\epsilon)} \cdot \sum_{i=0}^{k-1}\left(\left(1-\frac{1}{k}\right)\frac{1-\epsilon}{1+\epsilon}\right)^i$$
$$\geq \frac{(1-\epsilon)^2}{k(1+\epsilon)^2} \cdot \sum_{i=0}^{k-1}\left(\left(1-\frac{1}{k}\right)\left(\frac{1-\epsilon}{1+\epsilon}\right)^2\right)^i = \frac{1}{1+\frac{4k\epsilon}{(1-\epsilon)^2}}\left(1-\left(1-\frac{1}{k}\right)^k\left(\frac{1-\epsilon}{1+\epsilon}\right)^{2k}\right).$$

Particularly, when the objective function is submodular, the parameter ϵ in Definition 7 equals 0, and thus the approximation ratio of the GSEMO-C reaches the optimal one, $1-1/e$. When $\epsilon \leq 1/k$, we have

$$\frac{1}{1+\frac{2k\epsilon}{1-\epsilon}}\left(1-\left(1-\frac{1}{k}\right)^k\left(\frac{1-\epsilon}{1+\epsilon}\right)^k\right) \geq \frac{1}{1+\frac{2}{1-1/k}}\left(1-\frac{1}{e}\right) \geq \frac{1}{5}\left(1-\frac{1}{e}\right),$$

where the last inequality holds by $k \geq 2$; thus, the GSEMO-C can still achieve a constant approximation ratio, as shown below.

Corollary 6

For maximizing a monotone function f with a size constraint k, where f is ϵ-approximately submodular with $\epsilon \leq 1/k$, the expected running time of the GSEMO-C until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq k$ and $f(\boldsymbol{x}) \geq (1/5)(1-1/e) \cdot \mathrm{OPT}$ is $O(n^2(\log n + k))$.

6. Conclusion

This paper theoretically studies the approximation performance of EAs for solving the general classes of combinatorial optimization problems, i.e., maximizing submodular functions with/without a size constraint and maximizing monotone approximately submodular functions with a size constraint. We prove that within polynomial expected running time, a simple multi-objective EA called GSEMO-C can achieve good approximation guarantees for any concerned problem class. These results may help to provide a theoretical explanation for the empirically good performance of EAs in various applications. A question that will be examined in the future is whether simple single-objective EAs such as the (1+1)-EA can achieve good approximation guarantees on the concerned problem classes, which has been partially addressed recently [15]. It is also interesting to study the performance of EAs under more complicated constraints, e.g., matroid and knapsack constraints [31].

7. Acknowledgments

The authors want to thank the associate editor and anonymous reviewers for their helpful comments and suggestions. C. Qian, Y. Yu and K. Tang were supported by the NSFC (61603367, 61672478,

61876077). X. Yao was supported by the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (2017ZT07X386) and Shenzhen Peacock Plan (KQTD2016112514355531). Z.-H. Zhou was supported by the National Key R&D Program of China (2018YFB1004300) and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] A. A. Ageev and M. I. Sviridenko. An 0.828-approximation algorithm for the uncapacitated facility location problem. *Discrete Applied Mathematics*, 93(2):149–156, 1999.

[2] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific, Singapore, 2011.

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

[4] 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)*, pages 498–507, Sydney, Australia, 2017.

[5] C. Bian, C. Qian, and K. Tang. A general approach to running time analysis of multi-objective evolutionary algorithms. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 1405–1411, Stockholm, Sweden, 2018.

[6] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete algorithms (SODA'14)*, pages 1433–1452, Portland, OR, 2014.

[7] N. Buchbinder, M. Feldman, J. Seffi, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015.

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

[9] S. Droste, T. Jansen, and I. Wegener. A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with Boolean inputs. *Evolutionary Computation*, 6(2):185–196, 1998.

[10] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[11] E. R. Elenberg, R. Khanna, A. G. Dimakis, and S. Negahban. Restricted strong convexity implies weak submodularity. *Annals of Statistics*, 46(6B):3539–3568, 2018.

[12] U. Feige, V. S. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

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

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

[15] T. Friedrich, A. Göbel, F. Quinzan, and M. Wagner. Heavy-tailed mutation operators in single-objective combinatorial optimization. In *Proceedings of the 15th International Conference on Parallel Problem Solving from Nature (PPSN'18)*, pages 134–145, Coimbra, Portugal, 2018.

[16] T. Friedrich, A. Göbel, F. Neumann, F. Quinzan, and R. Rothenberger. Greedy maximization of functions with bounded curvature under partition matroid constraints. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*, Honolulu, HI, 2019.

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

[18] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[19] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[20] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

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

[22] T. Jansen and I. Wegener. Evolutionary algorithms – how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599, 2001.

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

[24] R. Khanna, E. R. Elenberg, A. G. Dimakis, J. Ghosh, and S. Negahban. On approximation guarantees for greedy low rank optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, pages 1837–1846, Sydney, Australia, 2017.

[25] J. R. Koza, M. A. Keane, and M. J. Streeter. What's AI done for me lately? Genetic programming's human-competitive results. *IEEE Intelligent Systems*, 18(3):25–31, 2003.

[26] A. Krause and V. Cevher. Submodular dictionary selection for sparse representation. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 567–574, Haifa, Israel, 2010.

[27] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI'05)*, pages 324–331, Edinburgh, Scotland, 2005.

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

[29] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

[30] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 16 (NIPS'03)*, pages 625–632, Vancouver, Canada, 2003.

[31] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09)*, pages 323–332, Bethesda, MD, 2009.

[32] G. Li and W. Chou. Path planning for mobile robot using self-adaptive learning particle swarm optimization. *Science China Information Sciences*, 61(5):052204, 2018.

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

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

[35] F. Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629, 2007.

[36] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[37] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40, 2007.

[38] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer-Verlag, Berlin, Germany, 2010.

[39] F. Neumann, J. Reichel, and M. Skutella. Computing minimum cuts by randomized search heuristics. *Algorithmica*, 59(3):323–342, 2011.

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

[41] C. Qian, Y. Yu, and Z.-H. Zhou. On constrained Boolean Pareto optimization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 389–395, Buenos Aires, Argentina, 2015.

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

[43] 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)*, pages 1478–1484, Stockholm, Sweden, 2018.

[44] J. Reichel and M. Skutella. Evolutionary algorithms and matroid optimization problems. *Algorithmica*, 57(1):187–206, 2010.

[45] M. C. Shewry and H. P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2): 165–170, 1987.

[46] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, pages 44–56, Stuttgart, Germany, 2005.

[47] Q. Yuan, H. Tang, W. You, X. Wang, and Y. Zhao. Virtual network function scheduling via multilayer encoding genetic algorithm with distributed bandwidth allocation. *Science China Information Sciences*, 61(9):092107, 2018.

[48] Z.-H. Zhou, Y. Yu, and C. Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, Singapore, 2019.