Quality-Diversity Algorithms Can Provably Be Helpful for Optimization

Chao Qian, **Ke Xue** and **Ren-Jian Wang**
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
School of Artificial Intelligence, Nanjing University, Nanjing 210023, China
{qianc, xuek, wangrj}@lamda.nju.edu.cn

Abstract

Quality-Diversity (QD) algorithms are a new type of Evolutionary Algorithms (EAs), aiming to find a set of high-performing, yet diverse solutions. They have found many successful applications in reinforcement learning and robotics, helping improve the robustness in complex environments. Furthermore, they often empirically find a better overall solution than traditional search algorithms which explicitly search for a single highest-performing solution. However, their theoretical analysis is far behind, leaving many fundamental questions unexplored. In this paper, we try to shed some light on the optimization ability of QD algorithms via rigorous runtime analysis. By comparing the popular QD algorithm MAP-Elites with $(\mu + 1)$-EA (a typical EA focusing on finding better objective values only), we prove that on two NP-hard problem classes with wide applications, i.e., monotone approximately submodular maximization with a size constraint, and set cover, MAP-Elites can achieve the (asymptotically) optimal polynomial-time approximation ratio, while $(\mu + 1)$-EA requires exponential expected time on some instances. This provides theoretical justification for that QD algorithms can be helpful for optimization, and discloses that the simultaneous search for high-performing solutions with diverse behaviors can provide stepping stones to good overall solutions and help avoid local optima.

1 Introduction

Search algorithms, e.g., simulated annealing, Evolutionary Algorithms (EAs), and Bayesian optimization, are traditionally to find a single highest-performing solution to the given problem. However, for many real-world tasks, e.g., in reinforcement learning [Eysenbach *et al.*, 2018; Chalumeau *et al.*, 2023], robotics [Cully *et al.*, 2015; Salehi *et al.*, 2022], and human-AI coordination [Lupu *et al.*, 2021; Strouse *et al.*, 2021], there is need for finding a set of high-performing solutions that exhibit varied behaviors. A behavior space can be characterized by dimensions of variation of interest to a user. For example, if searching for robot gaits that can run fast, the dimensions of interest can be the portion of time that each leg is in contact with the ground. Illuminating high-performing solutions of each area of the entire behavior space reveals relationships between dimensions of interest and performance, which can help improve the robustness in complex environments, e.g., help robots rapidly adapt to damage (i.e., discover a behavior compensating for the damage) [Cully *et al.*, 2015].

Quality-Diversity (QD) algorithms [Pugh *et al.*, 2016; Cully and Demiris, 2018] are a subset of EAs, and have emerged as a potent optimization paradigm for the challenging task of finding a set of high-quality and diverse solutions. They follow a typical evolutionary process, which iteratively select some parent solutions from the population (also called archive in the QD area), apply variation operators (e.g., crossover and mutation) to produce offspring solutions, and use these offspring solutions to update the population. Lehman and Stanley [2011b] proposed the QD algorithm, Novelty Search and Local Competition (NSLC), by maximizing a local competition objective (e.g., the number of nearest neighbors of a solution worse than itself) and a novelty objective (e.g., the average distance of a solution to its nearest neighbors) simultaneously. The Multi-Objective Landscape Exploration (MOLE) algorithm [Clune *et al.*, 2013] is similar to NSLC, except that a global performance objective, instead of the local competition objective, is considered.

Mouret and Clune [2015] proposed the Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) algorithm to achieve the goal of finding a set of high-quality and diverse solutions more straightforwardly. MAP-Elites maintains an archive by discretizing the behavior space into cells and storing at most one solution in each cell. The generated solutions compete only when their behaviors belong to the same cell, and MAP-Elites tries to fill in the cells with as high-quality solutions as possible. MAP-Elites performs better than NSLC and MOLE empirically [Mouret and Clune, 2015], and has become the most popular QD algorithm. Many advanced variants have been proposed to improve its sample efficiency, e.g., refining parent selection strategies [Cully and Demiris, 2018; Sfikas *et al.*, 2021; Wang *et al.*, 2022; Wang *et al.*, 2023] and variation operators [Nilsson and Cully, 2021; Tjanaka *et al.*, 2022; Faldor *et al.*, 2023], improving archive structure [Vassiliades *et al.*, 2018; Grillotti and Cully, 2022], using surrogate models [Lim *et al.*, 2022; Bhatt *et al.*, 2022], and applying cooperative coevolution [Xue *et al.*, 2024].

They have achieved encouraging performance in various reinforcement learning tasks, e.g., exploration [Conti *et al.*, 2018; Ecoffet *et al.*, 2021], robust training [Kumar *et al.*, 2020], and environment generation [Fontaine *et al.*, 2021; Zhang *et al.*, 2023].

Besides generating a diverse set of high-performing solutions, extensive experiments (e.g., on evolving locomoting virtual creatures [Lehman and Stanley, 2011b], producing modular neural networks [Mouret and Clune, 2015], designing soft robots [Mouret and Clune, 2015], encouraging exploration of agents in deceptive [Conti *et al.*, 2018] and sparse [Colas *et al.*, 2020] reward environments, and solving traveling thief problems [Nikfarjam *et al.*, 2022a]) have shown that QD algorithms can find a better overall solution than traditional search algorithms which focus on finding the single highest-performing solution. That is, if only looking at the best achieved objective value, QD algorithms are still better. Even when searching for behavioral novelty only while ignoring the objective, the empirical results on maze navigation and biped walking show that it significantly outperforms objective-based search [Lehman and Stanley, 2011a]. The better optimization ability of QD algorithms may be due to the better exploration of the search space, which helps avoid local optima and thus find better objective values.

Compared to their wide application, the theoretical analysis of QD algorithms is far behind. To the best of our knowledge, there have existed only two pieces of works on runtime analysis of QD algorithms. Nikfarjam et al. [2022b] analyzed the MAP-Elites algorithm for solving the knapsack problem. They proved that by using carefully designed two-dimensional behavior spaces, MAP-Elites can simulate dynamic programming behaviors to find an optimal solution within expected pseudo-polynomial time. Bossek and Sudholt [2023] further proved that for maximizing monotone submodular functions with a size constraint, MAP-Elites using the number of items contained by a set as the behavior descriptor can achieve a $(1-1/e)$-approximation ratio in expected polynomial time; by using the number of connected components as the behavior descriptor, MAP-Elites can solve the minimum spanning tree problem in expected polynomial time. They also analyzed MAP-Elites for solving any pseudo-Boolean problem $f : \{0,1\}^n \to \mathbb{R}$. The number of 1-bits of a solution is used as the behavior descriptor, and the i-th cell of the behavior space stores the best found solution with the number of 1-bits belonging to $[(i-1)k, ik-1]$, where k (dividing $n+1$) is a granularity parameter. They proved the upper bound on the expected runtime of MAP-Elites until covering all cells. For the specific pseudo-Boolean problem OneMax which aims to maximize the number of 1-bits of a solution, they proved that the derived bound is tight, even for the time to find the optimal solution within each cell. There are also some preliminary studies trying to understand the behavior of novelty search from a theoretical perspective. Doncieux et al. [2019] hypothesized that novelty search asymptotically behaves like uniform sampling in the behavior space, and Wiegand [2024] showed that novelty search will converge in the archive space by utilizing k nearest neighbor theory.

In this paper, we try to theoretically examine the fundamental question: whether QD algorithms can be helpful for optimization, i.e., finding better objective values? For this purpose, we compare the popular QD algorithm MAP-Elites with $(\mu+1)$-EA via rigorous runtime analysis. $(\mu+1)$-EA maintains a population of μ solutions, and focuses on finding better objective values by iteratively generating an offspring solution via mutation and using it to replace the worst solution in the population if it is better. For fair comparison, the population size μ of $(\mu+1)$-EA is set to the number of cells of the archive of MAP-Elites, and all their parameters are set to the same. The only difference is that for MAP-Elites, the generated offspring solution only competes with the solution in the same cell of the archive, while for $(\mu+1)$-EA, the offspring competes with all solutions in the population, and replaces the worst one if it is better. Thus, their performance gap can be purely attributed to the QD algorithms' framework of searching for a set of high-quality and diverse solutions.

We consider two NP-hard problem classes with wide applications. For monotone approximately submodular maximization with a size constraint [Das and Kempe, 2011], we prove that MAP-Elites achieves the optimal polynomial-time approximation ratio of $1-e^{-\gamma}$ [Harshaw *et al.*, 2019], where $\gamma \in [0,1]$ measures the closeness of a set function f to submodularity (**Theorem 1**). When f is submodular, $\gamma = 1$, and the approximation ratio of MAP-Elites is $1-1/e$ (**Corollary 1**); while there exists an instance where $(\mu+1)$-EA requires expected exponential time to achieve a nearly $1/2$-approximation ratio (**Theorem 2**). For the set cover problem with m elements [Feige, 1998], MAP-Elites achieves the asymptotically optimal approximation ratio of $\ln m + 1$ in expected polynomial time (**Theorem 3**), while $(\mu+1)$-EA requires expected exponential time to achieve an approximation ratio smaller than $2^{m+1}/m$ on some instance (**Theorem 4**). These results clearly show the advantage of MAP-Elites over $(\mu+1)$-EA, providing theoretical justification for that QD algorithms can be helpful for optimization. The analyses disclose that the simultaneous search of QD algorithms for high-performing solutions at diverse regions in the behavior space plays a key role, which provides stepping stones to high-performing solutions in a region that may be difficult to be discovered if searching only in that region (which may get trapped in local optima). *Due to space limitation, we only give the proof sketch of the theorems in the paper, and the proof details are given in the supplementary.*

2 Preliminaries

2.1 MAP-Elites Algorithm

MAP-Elites is the most popular QD algorithm [Mouret and Clune, 2015; Cully *et al.*, 2015]. Given an objective function $f : \mathcal{X} \to \mathbb{R}$ to be maximized, MAP-Elites tries to find a set of high-quality (i.e., having large f values) and diverse solutions, instead of a single highest-performing solution (i.e., a solution having the largest f value). Specifically, a user chooses m dimensions of variation of interest that define a behavior space, and MAP-Elites aims to create a map of high-performing solutions at each point in the behavior space. The behavior space can be characterized by a behavior descriptor function $\boldsymbol{b} : \mathcal{X} \to \mathbb{R}^m$. For a solution $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{b}(\boldsymbol{x})$ returns an m-dimensional vector describing $\boldsymbol{x}$'s behaviors.

Algorithm 1 MAP-Elites Algorithm

Input: an objective function $f : \mathcal{X} \to \mathbb{R}$, and a behavior descriptor function $\boldsymbol{b} : \mathcal{X} \to \mathbb{R}^m$
Parameter: #initial solutions I

1: Initialize an empty archive $M \leftarrow \emptyset$, and let $iter = 0$;
2: **repeat**
3: **if** $iter < I$ **then**
4: Choose $\boldsymbol{x}'$ from $\mathcal{X}$ uniformly at random
5: **else**
6: Choose $\boldsymbol{x}$ from M uniformly at random;
7: Create $\boldsymbol{x}'$ by mutating $\boldsymbol{x}$
8: **end if**
9: **if** $M(\boldsymbol{b}(\boldsymbol{x}')) = \emptyset$ or $f(\boldsymbol{x}') > f(M(\boldsymbol{b}(\boldsymbol{x}')))$ **then**
10: $M(\boldsymbol{b}(\boldsymbol{x}')) = \boldsymbol{x}'$
11: **end if**
12: $iter = iter + 1$
13: **until** some criterion is met
14: **return** M

In our theoretical analysis, we use the simple default version of MAP-Elites [Mouret and Clune, 2015], as presented in Algorithm 1. For each behavior descriptor vector $\boldsymbol{b}(\cdot)$, an archive M uses a cell (denoted as $M(\boldsymbol{b}(\cdot))$) to store the highest-performing solution found so far. Note that multiple solutions may have the same behavior descriptor vector. If no solution is found for a cell in the archive M, the cell is empty. The archive M is initialized to be empty in line 1. The first I solutions are randomly generated (lines 3–4). After that, a solution $\boldsymbol{x}$ is randomly chosen from the archive M (line 6), and mutated to generate an offspring solution $\boldsymbol{x}'$ (line 7) in each iteration. Each generated solution $\boldsymbol{x}'$ will be used to update the corresponding cell $M(\boldsymbol{b}(\boldsymbol{x}'))$, as shown in lines 9–11. If the cell is empty (i.e., $M(\boldsymbol{b}(\boldsymbol{x}')) = \emptyset$), $\boldsymbol{x}'$ will be directly placed there. If $\boldsymbol{x}'$ is higher-performing than the current occupant of the cell (i.e., $f(\boldsymbol{x}') > f(M(\boldsymbol{b}(\boldsymbol{x}')))$), the occupant will be replaced by $\boldsymbol{x}'$. Note that according to user preference or available computational resources, multiple behavior descriptor vectors may be merged into one cell in practice.

2.2 $(\mu + 1)$-EA

$(\mu+1)$-EA is a typical EA focusing on finding solutions with larger f values, which has been widely used in theoretical analyses of EAs, e.g., [Jansen and Wegener, 2001; Witt, 2006; Storch, 2008]. As presented in Algorithm 2, $(\mu+1)$-EA maintains a set of μ solutions, i.e., employs a population size of μ. It first initializes a population P by randomly generating μ solutions in line 1. Then, in each iteration, it generates one new solution $\boldsymbol{x}'$ by mutating a solution $\boldsymbol{x}$ randomly chosen from the population P (lines 3–4), and uses $\boldsymbol{x}'$ to replace the worst solution in P if $\boldsymbol{x}'$ is better (lines 5–8).

2.3 Algorithm Settings & Performance Measure

To theoretically show that QD algorithms can be helpful for optimization (i.e., finding a single, best solution), we will compare MAP-Elites with $(\mu+1)$-EA. As the problems studied in Sections 3 and 4 have the solution space $\mathcal{X} = \{0,1\}^n$ (where n denotes the problem size), we employ the commonly used *bit-wise mutation* operator in Algorithms 1 and 2,

Algorithm 2 $(\mu + 1)$-EA

Input: an objective function $f : \mathcal{X} \to \mathbb{R}$
Parameter: population size μ

1: Initialize a population P by choosing μ solutions from $\mathcal{X}$ uniformly at random;
2: **repeat**
3: Choose $\boldsymbol{x}$ from P uniformly at random;
4: Create $\boldsymbol{x}'$ by mutating $\boldsymbol{x}$;
5: Let $\boldsymbol{y} = \arg\min_{\boldsymbol{y} \in P} f(\boldsymbol{y})$; ties are broken randomly;
6: **if** $f(\boldsymbol{x}') > f(\boldsymbol{y})$ **then**
7: $P \leftarrow (P \setminus \{\boldsymbol{y}\}) \cup \{\boldsymbol{x}'\}$
8: **end if**
9: **until** some criterion is met
10: **return** P

which flips each bit of a solution $\boldsymbol{x}$ with probability $1/n$ to generate a new solution $\boldsymbol{x}'$. For fairness, the population size μ of $(\mu + 1)$-EA is set to the number of cells in the archive M; as $(\mu + 1)$-EA generates μ initial solutions randomly, the number of randomly generated initial solutions (i.e., the parameter I) of MAP-Elites is set equal to μ. These settings are to make that the performance gap between MAP-Elites and $(\mu+1)$-EA is purely due to the mechanism of finding a set of high-quality and diverse solutions of MAP-Elites.

To be consistent with the original setting of MAP-Elites in [Mouret and Clune, 2015], we have used strict survivor selection, i.e., $f(\boldsymbol{x}') > f(M(\boldsymbol{b}(\boldsymbol{x}')))$ in line 9 of Algorithm 1 and $f(\boldsymbol{x}') > f(\boldsymbol{y})$ in line 6 of Algorithm 2. However, our analyses will not be affected if allowing accepting the same good solutions, i.e., replacing $>$ with $\geq$. We consider maximization problems in Algorithms 1 and 2. For minimization, we only need slight modification, i.e., replacing $>$ with $<$ in both algorithms, and changing $\boldsymbol{y} = \arg\min_{\boldsymbol{y} \in P} f(\boldsymbol{y})$ in line 5 of Algorithm 2 to $\boldsymbol{y} = \arg\max_{\boldsymbol{y} \in P} f(\boldsymbol{y})$.

To measure the performance of QD algorithms, the three metrics, i.e., *optimization*, *coverage* and *QD-Score*, are often used [Pugh *et al.*, 2016; Cully and Demiris, 2018]. Optimization corresponds to the largest objective value of solutions in the archive, i.e., the objective value of the highest-performing solution found by the algorithm. Coverage is the number of non-empty cells, i.e., the total number of solutions, in the archive. QD-Score is equal to the total sum of objective values across all solutions in the archive, which reflects both the quality and diversity of the found solutions.

Here, we focus on the optimization metric, serving for our purpose of showing that QD algorithms can be helpful for optimization, i.e., finding better objective values. Specifically, we will analyze the expected runtime complexity of MAP-Elites or $(\mu + 1)$-EA, which counts the number of objective evaluations (the most time-consuming step in the evolutionary process) until finding an α-approximate solution, and has been a leading theoretical aspect for randomized search heuristics [Neumann and Witt, 2010; Auger and Doerr, 2011; Zhou *et al.*, 2019; Doerr and Neumann, 2020]. Let OPT denote the optimal function value. An α-approximate solution $\boldsymbol{x}$ means $f(\boldsymbol{x}) \geq \alpha \cdot \mathrm{OPT}$ (where $\alpha \leq 1$) for maximization, and $f(\boldsymbol{x}) \leq \alpha \cdot \mathrm{OPT}$ (where $\alpha \geq 1$) for minimization.

3 Monotone Approximately Submodular Maximization with a Size Constraint

Let $\mathbb{R}$ and $\mathbb{R}^+$ denote the set of reals and non-negative reals, respectively. Given a finite set $V = \{v_1, \ldots, v_n\}$, a set function $f : 2^V \to \mathbb{R}$ is monotone if $\forall X \subseteq Y, f(X) \leq f(Y)$. We assume w.l.o.g. that monotone functions are normalized, i.e., $f(\emptyset) = 0$. A set function f is submodular [Nemhauser *et al.*, 1978] if $\forall X \subseteq Y \subseteq V, f(Y) - f(X) \leq \sum_{v \in Y \setminus X} \big(f(X \cup \{v\}) - f(X)\big)$. The submodularity ratio in Definition 1 measures how close a general set function f is to submodularity. When f is monotone, it holds: $\forall X, l : 0 \leq \gamma_{X,l}(f) \leq 1$; f is submodular iff $\forall X, l : \gamma_{X,l}(f) = 1$.

Definition 1 (Submodularity Ratio [Das and Kempe, 2011]). *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*

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

As presented in Definition 2, the studied problem is to find a subset with at most k items such that a given monotone and approximately submodular objective function is maximized. It is NP-hard, and has many applications, such as maximum coverage [Feige, 1998], influence maximization [Kempe *et al.*, 2003], sensor placement [Krause *et al.*, 2008], sparse regression [Das and Kempe, 2011], and unsupervised feature selection [Feng *et al.*, 2019], just to name a few.

Definition 2 (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, the goal is to find a subset $X \subseteq V$ such that*

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

In the following analysis, we equivalently reformulate the above problem as an unconstrained maximization problem by setting $f(X) = -1$ when $|X| > k$, i.e., X does not satisfy the size constraint. Note that a subset X of V can be naturally represented by a Boolean vector $\boldsymbol{x} \in \{0,1\}^n$, where the i-th bit $x_i = 1$ if $v_i \in X$, and $x_i = 0$ otherwise. Throughout the paper, we will not distinguish $\boldsymbol{x} \in \{0,1\}^n$ and its corresponding subset $\{v_i \in V \mid x_i = 1\}$ for convenience.

3.1 Analysis of MAP-Elites

To apply MAP-Elites in Algorithm 1 to maximize monotone approximately submodular functions with a size constraint, we use the number of 1-bits (denoted as $|\boldsymbol{x}|$, which equals $\sum_{i=1}^n x_i$) of a solution $\boldsymbol{x} \in \{0,1\}^n$ as the behavior descriptor. Thus, the archive M maintained by MAP-Elites contains $n+1$ cells, where for any $0 \leq i \leq n$, the i-th cell stores the best found solution with i 1-bits. In each iteration of MAP-Elites, if an offspring solution $\boldsymbol{x}'$ with j 1-bits is generated, the j-th cell in the archive M will be examined: if the cell is empty or the existing solution in the cell is worse than $\boldsymbol{x}'$, the j-th cell will then be occupied by $\boldsymbol{x}'$.

We prove in Theorem 1 that after running $O(n^2(\log n + k))$ expected time, MAP-Elites achieves an approximation ratio of $1 - e^{-\gamma_{\min}}$, which reaches the optimal polynomial-time approximation ratio [Harshaw *et al.*, 2019]. Inspired from the analysis of GSEMO (a multi-objective EA with similar procedure to MAP-Elites) [Friedrich and Neumann, 2015; Qian *et al.*, 2015b; Qian *et al.*, 2019], the proof can be accomplished by following the behavior of the greedy algorithm [Das and Kempe, 2011]. Note that the parameter $I = n+1$ implies that MAP-Elites generates $n+1$ initial solutions randomly. The value of I does not affect the analysis, and such a setting is just for fairness, because the population size μ of $(\mu+1)$-EA will be set to the number of cells (i.e., $n+1$) contained by the archive M.

Theorem 1. *For maximizing a monotone approximately submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n+1$, until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| = k$ 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.*

Proof Sketch. We divide the optimization process of MAP-Elites into two phases: (1) starts after initialization and finishes after finding the special solution $\mathbf{0}$ (i.e., the empty set); (2) starts after phase (1) and finishes after finding a solution with the desired approximation guarantee.

For phase (1), we consider the minimum number of 1-bits of the solutions in the archive M, denoted as $J_{\min}$. We show that $J_{\min}$ will not increase, but can decrease in each iteration by selecting the solution $\boldsymbol{x}$ corresponding to $J_{\min}$ and flipping only one 1-bit in lines 6–7 of Algorithm 1, which happens with probability $(1/|M|) \cdot (J_{\min}/n)(1-1/n)^{n-1} \geq J_{\min}/(en(n+1))$, where $|M| \leq n+1$. Thus, the expected number of iterations required by phase (1) (i.e., making $J_{\min} = 0$) is at most $\sum_{i=1}^n en(n+1)/i = O(n^2 \log n)$.

For phase (2), we consider $J_{\max}$, which is defined as the maximum value of $j \in \{0, 1, \ldots, k\}$ such that in the archive M, there is a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| = j$ and $f(\boldsymbol{x}) \geq (1 - (1 - \gamma_{\min}/k)^j) \cdot \mathrm{OPT}$. We only need to analyze the expected number of iterations until $J_{\max} = k$, which implies that $\exists \boldsymbol{x} \in M$ with $|\boldsymbol{x}| = k$ and $f(\boldsymbol{x}) \geq (1 - (1 - \gamma_{\min}/k)^k) \cdot \mathrm{OPT} \geq (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$, i.e., the desired approximation guarantee is reached. We show that $J_{\max}$ will not decrease, but can increase in each iteration by selecting the solution $\boldsymbol{x}$ corresponding to $J_{\max}$ and flipping only one specific 0-bit (according to Lemma 1 of [Qian *et al.*, 2016]), which happens with probability $(1/|M|) \cdot (1/n)(1-1/n)^{n-1} \geq 1/(en(n+1))$. Thus, the expected number of iterations required by phase (2) (i.e., making $J_{\max} = k$) is at most $k \cdot en(n+1)$.

Summing up the expected time of the two phases leads to the theorem. Note that the additional evaluations for the $n+1$ initial solutions will not affect the derived time bound. □

For submodular f, $\forall \boldsymbol{x} : \gamma_{\boldsymbol{x},k} = 1$, and $\gamma_{\min} = 1$. Thus,

Corollary 1. *For maximizing a monotone submodular function f with a size constraint k, the expected runtime of MAP-Elites with $I = n+1$, until finding a solution $\boldsymbol{x}$ with $|\boldsymbol{x}| = k$ and $f(\boldsymbol{x}) \geq (1 - 1/e) \cdot \mathrm{OPT}$, is $O(n^2(\log n + k))$.*

Corollary 1 is consistent with Theorem 5.1 in [Bossek and Sudholt, 2023]. Note that the MAP-Elites algorithm they analyzed is slightly different from ours, which generates only one random initial solution (i.e., the parameter $I = 1$) and replaces the solution in a cell if the generated offspring belonging to the same cell is not worse (i.e., $f(\boldsymbol{x}') > f(M(\boldsymbol{b}(\boldsymbol{x}')))$

in line 9 of Algorithm 1 changes to $f(\boldsymbol{x}') \geq f(M(\boldsymbol{b}(\boldsymbol{x}')))$. Thus, these slight differences do not affect the result.

3.2 Analysis of $(\mu+1)$-EA

To apply $(\mu+1)$-EA in Algorithm 2 to maximize monotone approximately submodular functions with a size constraint, we set the population size μ to the number of cells contained by the archive M of MAP-Elites, which is $n+1$. As we have mentioned in Section 2.3, such a setting is for fairness, which makes the population size of $(\mu+1)$-EA equal to the largest archive size of MAP-Elites. In each iteration of $(\mu+1)$-EA, the generated offspring solution $\boldsymbol{x}'$ will be used to replace the worst solution in the current population if $\boldsymbol{x}'$ is better.

Next, we will give a concrete instance of the classic NP-hard problem, maximum coverage [Feige, 1998] (which is an application of monotone submodular maximization with a size constraint), where $(\mu+1)$-EA fails to achieve a good approximation ratio in polynomial time. Given a family V of sets that cover a universe U of elements, the goal of maximum coverage as presented in Definition 3 is to select at most k sets from V to make their union maximal. The objective function $f(\boldsymbol{x}) = |\bigcup_{i:x_i=1} S_i|$ is monotone and submodular.

Definition 3 (Maximum Coverage)**.** *Given a ground set U, a collection $V = \{S_1, S_2, \ldots, S_n\}$ of subsets of U, and a budget k, the goal is to find a subset $\boldsymbol{x}$ of V such that*

$$\arg\max_{\boldsymbol{x}\in\{0,1\}^n} \quad |\bigcup\nolimits_{i:x_i=1} S_i| \quad s.t. \quad |\boldsymbol{x}| \leq k.$$

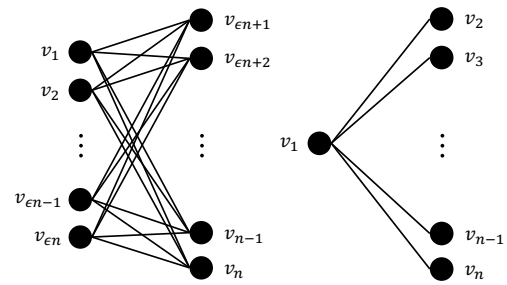
The considered instance of maximum coverage as presented in Example 1 is from [Friedrich *et al.*, 2010; Friedrich and Neumann, 2015], and can be obtained from the complete bipartite graph in Figure 1(a). The ground set U is given by the set of edges. Each vertex v_i corresponds to one subset S_i of U, which consists of the edges adjacent to the vertex v_i. The goal is to select at most $k = (1+\delta)n/3$ subsets of U to make the number of contained edges maximal.

Example 1. *The ground set U contains all the edges of the complete bipartite graph in Figure 1(a). Each set S_i in V consists of the edges adjacent to the vertex v_i, i.e., $\forall 1 \leq i \leq (1+\delta)n/3$, $S_i = \{(v_i, v_{(1+\delta)n/3+1}), \ldots, (v_i, v_n)\}$; $\forall (1+\delta)n/3+1 \leq i \leq n$, $S_i = \{(v_i, v_1), \ldots, (v_i, v_{(1+\delta)n/3})\}$, where (v_i, v_j) denotes the edge between the vertexes v_i and v_j. The budget $k = (1+\delta)n/3$, and δ is a small positive constant close to 0.*

The two parts of the complete bipartite graph in Figure 1(a) contain $(1+\delta)n/3$ and $(2-\delta)n/3$ vertexes, respectively. As the budget $k = (1+\delta)n/3 < (2-\delta)n/3$, the unique optimal solution is $\boldsymbol{x}^* = 1^{(1+\delta)n/3}0^{(2-\delta)n/3}$ (i.e., $\{S_1, \ldots, S_{(1+\delta)n/3}\}$), which covers all the edges, and has the objective value $f(\boldsymbol{x}^*) = (1+\delta)(2-\delta)n^2/9$. If a solution

$$\boldsymbol{x}_{\text{local}} \in \{0^{(1+\delta)n/3}\boldsymbol{y} \mid \boldsymbol{y} \in \{0,1\}^{(2-\delta)n/3} \wedge |\boldsymbol{y}| = (1+\delta)n/3\},$$

i.e., contains $(1+\delta)n/3$ sets from $\{S_{(1+\delta)n/3+1}, \ldots, S_n\}$, it is local optimal, with $f(\boldsymbol{x}_{\text{local}}) = (1+\delta)^2 n^2/9$. Adding a set from $\{S_1, \ldots, S_{(1+\delta)n/3}\}$ into $\boldsymbol{x}_{\text{local}}$ will cover $(2-\delta)n/3 - (1+\delta)n/3 = (1-2\delta)n/3$ more edges. If i sets from $\{S_1, \ldots, S_{(1+\delta)n/3}\}$ have been added into $\boldsymbol{x}_{\text{local}}$, then



(a) $\epsilon = (1+\delta)/3$ (b)

Figure 1: Two complete bipartite graphs.

deleting one set from $\boldsymbol{x}_{\text{local}}$ will decrease the number of edges by $(1+\delta)n/3 - i$. To make $(1+\delta)n/3 - i \leq (1-2\delta)n/3$, i should be at least δn. Thus, to improve the local optimal solution $\boldsymbol{x}_{\text{local}}$, at least δn sets from $\{S_1, \ldots, S_{(1+\delta)n/3}\}$ need to be added into $\boldsymbol{x}_{\text{local}}$, and meanwhile at least δn sets from $\boldsymbol{x}_{\text{local}}$ have to be deleted. The approximation ratio of $\boldsymbol{x}_{\text{local}}$ is $f(\boldsymbol{x}_{\text{local}})/f(\boldsymbol{x}^*) = (1+\delta)/(2-\delta)$.

We prove in Theorem 2 that $(\mu+1)$-EA requires at least exponential (w.r.t. n) expected runtime to achieve an approximation ratio larger than $(1+\delta)/(2-\delta)$ on Example 1. The main proof idea is to show that there is some probability that the population of $(\mu+1)$-EA will get trapped in a local optimal solution $\boldsymbol{x}_{\text{local}}$, escaping from which requires at least $2\delta n$ bits to be flipped simultaneously.

Theorem 2. *There exists an instance of monotone submodular maximization with a size constraint (i.e., the maximum coverage instance in Example 1), where the expected runtime of $(\mu+1)$-EA with $\mu = n+1$ for achieving an approximation ratio larger than $(1+\delta)/(2-\delta)$ is at least exponential w.r.t. n, where δ is a small positive constant close to 0.*

Proof Sketch. $(\mu+1)$-EA generates $n+1$ initial solutions by sampling from $\{0,1\}^n$ uniformly at random. We first consider the event A that among the generated $n+1$ initial solutions, one is a local optimal solution $\boldsymbol{x}_{\text{local}}$, and the other n solutions do not satisfy the size constraint, i.e., contain more than $k = (1+\delta)n/3$ 1-bits. It happens with probability $P(A) \geq (n+1)(1-o(1))/2^n$ by the Chernoff bound.

Given the event A, we consider that in the first n iterations of $(\mu+1)$-EA, a local optimal solution is always selected from the population in line 3 of Algorithm 2, and no bits are flipped by the bit-wise mutation operator in line 4. Such an event (denoted as B) will make the population full of local optimal solutions, because the copied local optimal solution will replace one solution violating the size constraint (whose objective value is set to -1) in the population after each iteration. It happens with probability $P(B) \geq \sqrt{2\pi n}/(e(2e^2)^n)$.

To improve $\boldsymbol{x}_{\text{local}}$, it needs to flip at least δn of its $(1+\delta)n/3$ leading 0-bits and δn of its $(1+\delta)n/3$ 1-bits, with probability at most $\binom{(1+\delta)n/3}{\delta n}\binom{(1+\delta)n/3}{\delta n}\left(\frac{1}{n}\right)^{2\delta n} \leq \frac{1}{2\pi\delta n}\left(\frac{e(1+\delta)}{3\delta n}\right)^{2\delta n}$. This implies that the expected runtime to generate a solution better than $\boldsymbol{x}_{\text{local}}$ is at least $2\pi\delta n(3\delta n/(e(1+\delta)))^{2\delta n}$.

Combining the above analysis and using the fact that the approximation ratio of $\boldsymbol{x}_{\text{local}}$ is $(1+\delta)/(2-\delta)$, we derive

that the expected runtime of $(\mu+1)$-EA for achieving an approximation ratio larger than $(1+\delta)/(2-\delta)$ is at least $P(A)\cdot P(B)\cdot 2\pi\delta n(3\delta n/(e(1+\delta)))^{2\delta n} \geq ((n+1)(1-o(1))/2^n)\cdot(\sqrt{2\pi n}/(e(2e^2)^n))\cdot 2\pi\delta n(3\delta n/(e(1+\delta)))^{2\delta n}$, which is exponential w.r.t. n because δ is a small positive constant close to 0. Thus, the theorem holds. □

The comparison between Corollary 1 and Theorem 2 clearly shows the advantage of MAP-Elites over $(\mu+1)$-EA. That is, for monotone submodular maximization with a size constraint, MAP-Elites can always achieve the $(1-1/e)$-approximation ratio in expected polynomial time, while there exists an instance where $(\mu+1)$-EA requires at least expected exponential time to achieve an approximation ratio of nearly $1/2$. From the proofs, we can find that MAP-Elites explicitly maintains multiple cells of the behavior space, allowing it to achieve a good solution with k 1-bits by following a path across different cells (i.e., gradually adding the number of 1-bits), while $(\mu+1)$-EA only focuses on the objective value, and may lose diversity in the population and get trapped in local optima (with k 1-bits). This discloses the benefit of simultaneously searching for high-performing solutions at diverse regions in the behavior space, which may provide stepping stones to high-performing solutions in a region that may be difficult to be discovered if searching only in that region.

4 Set Cover

Set cover [Feige, 1998] is a well-known NP-hard combinatorial optimization problem. Given a family V of sets (where each set has a corresponding positive weight) that cover a universe U of elements, the goal is to find a subset of V with the minimum weight such that all the elements of U are covered.

Definition 4 (Set Cover)**.** *Given a ground set $U = \{e_1,\ldots,e_m\}$, and a collection $V=\{S_1,S_2,\ldots,S_n\}$ of subsets of U with corresponding weights $w: V\to\mathbb{R}^+$, the goal is to find a subset of V (represented by $\boldsymbol{x}\in\{0,1\}^n$) such that*

$$\arg\min_{x\in\{0,1\}^n} \sum_{i=1}^n w_i x_i \quad \textit{s.t.} \quad \bigcup_{i:x_i=1} S_i = U.$$

Let $w_{\max}=\max\{w_i \mid 1\leq i\leq n\}$ and $w_{\min}=\min\{w_i \mid 1\leq i\leq n\}$ denote the maximum and minimum weights of a single set S_i, respectively. In the following analysis, we equivalently reformulate the above problem as an unconstrained minimization problem by setting

$$\forall \boldsymbol{x}\in\{0,1\}^n : f(\boldsymbol{x}) = w(\boldsymbol{x}) + \lambda\cdot(m-c(\boldsymbol{x})), \quad (1)$$

where $w(\boldsymbol{x})=\sum_{i=1}^n w_i x_i$ denotes the weight of $\boldsymbol{x}$, $c(\boldsymbol{x}) = |\bigcup_{i:x_i=1} S_i|$ denotes the number of elements covered by $\boldsymbol{x}$, and $\lambda > n w_{\max}$ is a sufficiently large penalty. That is, a solution covering more elements is better; if covering the same number of elements, a solution with a smaller weight is better.

4.1 Analysis of MAP-Elites

To apply MAP-Elites in Algorithm 1 to solve the set cover problem, we use the number $c(\boldsymbol{x}) = |\bigcup_{i:x_i=1} S_i|$ of covered elements of a solution $\boldsymbol{x}\in\{0,1\}^n$ as the behavior descriptor. Thus, the archive M maintained by MAP-Elites contains $m+1$ cells, where for any $0\leq i\leq m$, the i-th cell stores the best found solution covering i elements. In each iteration of MAP-Elites, if an offspring solution $\boldsymbol{x}'$ with $c(\boldsymbol{x}')=j$ is generated, the j-th cell in M will be examined: if the cell is empty or the existing solution in the cell is worse than $\boldsymbol{x}'$ (i.e., has a larger weight than $\boldsymbol{x}'$), the j-th cell will then be occupied by $\boldsymbol{x}'$.

We prove in Theorem 3 that after running $O(mn(m+\log n+\log(w_{\max}/w_{\min})))$ expected time, MAP-Elites achieves an approximation ratio of $\ln m+1$, which has been optimal up to a constant factor, unless P = NP [Feige, 1998]. The proof is accomplished by following the behavior of the greedy algorithm [Chvatal, 1979], inspired from the analysis of GSEMO [Friedrich *et al.*, 2010; Qian *et al.*, 2015a]. The parameter I of MAP-Elites is set to $m+1$ (i.e., it generates $m+1$ initial solutions randomly) for fairness, because the population size μ of $(\mu+1)$-EA will be set to the number of cells (i.e., $m+1$) contained by the archive M.

Theorem 3. *For the set cover problem, the expected runtime of MAP-Elites with the parameter $I=m+1$, until finding a solution $\boldsymbol{x}$ with $c(\boldsymbol{x})=m$ and $f(\boldsymbol{x})\leq(\ln m+1)\cdot\mathrm{OPT}$, is $O(mn(m+\log n+\log(w_{\max}/w_{\min})))$, where $c(\boldsymbol{x}) = |\bigcup_{i:x_i=1} S_i|$ denotes the number of elements covered by $\boldsymbol{x}$.*

Proof Sketch. We divide the optimization process of MAP-Elites into two phases. Phase (1) starts after initialization and finishes after finding the special solution $\mathbf{0}$ which does not cover any element. We consider the minimum weight of solutions in the archive M after running t iterations, denoted as $X_t=\min\{w(\boldsymbol{x}) \mid \boldsymbol{x}\in M\}$. Note that $X_t=0$ implies that the solution $\mathbf{0}$ has been found. By showing that the expected change of X_t after one iteration satisfies $E(X_t - X_{t+1} \mid X_t) \geq X_t/(en(m+1))$ and applying the multiplicative drift analysis theorem [Doerr *et al.*, 2012], we derive that the expected runtime to make $X_t=0$ (i.e., find the solution $\mathbf{0}$) is at most $O(mn(\log n+\log(w_{\max}/w_{\min})))$.

After that, phase (2) aims to find a solution with the desired approximation guarantee. Let $H_i=\sum_{j=1}^i 1/j$ denote the i-th harmonic number, and $R_k = H_m - H_{m-k}$. We consider $K_{\max}$, which is the maximum value of k such that $\exists\boldsymbol{x}\in M$ with $c(\boldsymbol{x})=k$ and $w(\boldsymbol{x})\leq R_k\cdot\mathrm{OPT}$. Then, we only need to analyze the expected runtime until $K_{\max}=m$, implying having found a solution with $c(\boldsymbol{x})=m$ (i.e., covering all the elements) and $w(\boldsymbol{x})\leq R_m\cdot\mathrm{OPT} < (\ln m+1)\cdot\mathrm{OPT}$. We show that $K_{\max}$ will not decrease, but can increase by flipping a specific 0 bit (i.e., adding the set S_i with the minimum ratio of its weight and the number of elements it newly covered) of the solution $\boldsymbol{x}$ corresponding to $K_{\max}$, which happens with probability $(1/|M|)\cdot(1/n)(1-1/n)^{n-1}\geq 1/(en(m+1))$, where $|M|\leq m+1$. Since it is sufficient to increase $K_{\max}$ at most m times for making $K_{\max}=m$, the expected runtime of phase (2) is $O(m^2 n)$. By combining the two phases, the total expected runtime is $O(mn(m+\log n+\log(w_{\max}/w_{\min})))$. Thus, the theorem holds. □

4.2 Analysis of $(\mu+1)$-EA

Next, we will give a concrete instance of set cover, where $(\mu+1)$-EA fails to achieve a good approximation ratio in polynomial time. To apply $(\mu+1)$-EA in Algorithm 2 to solve the set cover problem, the population size μ is set to the number of cells (i.e., $m+1$) contained by the archive M of

MAP-Elites for fair comparison. The considered instance is obtained from the complete bipartite graph in Figure 1(b).

Example 2. *The ground set U contains all the $n-1$ edges of the complete bipartite graph in Figure 1(b). Each set S_i in V consists of the edges adjacent to the vertex v_i, i.e., $S_1 = \{(v_1, v_2), \ldots, (v_1, v_n)\}$; $\forall 2 \le i \le n$, $S_i = \{(v_i, v_1)\}$, where (v_i, v_j) denotes the edge between the vertexes v_i and v_j. Their weights are: $w_1 = 2^n$; $\forall 2 \le i \le n$, $w_i = 1$.*

In this example, the size m of the ground set U is $n-1$, $w_{\max} = 2^n$, and $w_{\min} = 1$. The unique optimal solution is $\boldsymbol{x}^* = 01^{n-1}$ (i.e., $\{S_2, \ldots, S_n\}$), which covers all the edges, and has the minimum weight $w(\boldsymbol{x}^*) = n-1$. We can also observe that $\boldsymbol{x}_{\text{local}} = 10^{n-1}$ (i.e., containing only S_1) is local optimal, with the weight $w(\boldsymbol{x}_{\text{local}}) = 2^n$. Furthermore, $\boldsymbol{x}_{\text{local}}$ has the smallest weight except $\boldsymbol{x}^*$, implying that it needs to flip all the n bits to improve $\boldsymbol{x}_{\text{local}}$. Its approximation ratio is $w(\boldsymbol{x}_{\text{local}})/w(\boldsymbol{x}^*) = 2^n/(n-1) = 2^{m+1}/m$. Note that if a solution does not cover all the edges, i.e., does not satisfy the constraint, its objective value will be penalized, as shown by the reformulated equivalent objective function in Eq. (1).

We prove in Theorem 4 that $(\mu+1)$-EA requires at least exponential (w.r.t. m, n, and $\log(w_{\max}/w_{\min})$) expected time to achieve an approximation ratio smaller than $2^{m+1}/m$ on Example 2. The proof idea is similar to that of Theorem 2, i.e., to show that there is some probability that the population will get trapped in the local optimum $\boldsymbol{x}_{\text{local}}$, escaping from which requires all the n bits to be flipped simultaneously.

Theorem 4. *There is a set cover instance (i.e., Example 2), where the expected runtime of $(\mu+1)$-EA with $\mu = m+1$ for achieving an approximation ratio smaller than $2^{m+1}/m$ is at least exponential w.r.t. m, n, and $\log(w_{\max}/w_{\min})$.*

Proof Sketch. As the population size $\mu = m+1 = n$, $(\mu+1)$-EA generates n initial solutions randomly by uniform sampling from $\{0,1\}^n$. We consider the event A that one initial solution is the local optimal solution $\boldsymbol{x}_{\text{local}}$, and the other $n-1$ initial solutions are all worse than $\boldsymbol{x}_{\text{local}}$, which happens with probability $P(A) \ge n(1-o(1))/2^n$. Given the event A, we then consider that in the first $n-1$ iterations of $(\mu+1)$-EA, $\boldsymbol{x}_{\text{local}}$ is always selected from the population, and no bits are flipped by bit-wise mutation. This event (denoted as B) will make the population consist of n copies of $\boldsymbol{x}_{\text{local}}$, and happens with probability $P(B) \ge \sqrt{2\pi(n-1)}/(e(2e^2)^{n-1})$.

After that, it needs to flip all the n bits of $\boldsymbol{x}_{\text{local}}$ to make an improvement, whose probability is $1/n^n$. Because the approximation ratio of $\boldsymbol{x}_{\text{local}}$ is $2^n/(n-1)$, the expected runtime of $(\mu+1)$-EA for achieving an approximation ratio smaller than $2^n/(n-1)$ is at least $P(A) \cdot P(B) \cdot n^n \ge (n(1-o(1))/2^n) \cdot (\sqrt{2\pi(n-1)}/(e(2e^2)^{n-1})) \cdot n^n$, which is exponential w.r.t. n. As $m = n-1$ and $\log(w_{\max}/w_{\min}) = \log(2^n/1) = n$ in Example 2, the theorem holds. □

Theorems 3 and 4 show that MAP-Elites is superior to $(\mu+1)$-EA for set cover, because MAP-Elites can always achieve the $(\ln m + 1)$-approximation ratio in expected polynomial time, while $(\mu+1)$-EA requires expected exponential time to achieve an approximation ratio smaller than $2^{m+1}/m$ on some instance. As found in the last section, the simultaneous search for high-performing solutions at diverse regions in the behavior space makes MAP-Elites better. More specifically, MAP-Elites obtains a good solution covering all elements by following a path across different cells of the behavior space (i.e., gradually covering more elements), while $(\mu+1)$-EA may lose diversity in the population and get trapped in local optima in the space of solutions covering all elements.

5 Conclusion and Discussion

This paper contributes to providing theoretical justification for the benefit of QD algorithms, i.e., bringing better optimization. On the two NP-hard problems, monotone approximately submodular maximization with a size constraint, and set cover, we prove that MAP-Elites achieves the (asymptotically) optimal polynomial-time approximation ratio, while $(\mu+1)$-EA requires exponential expected time on some instances. The simultaneous search of QD algorithms for high-performing solutions with diverse behaviors brings better exploration of the search space, helping avoid local optima and find better solutions. In fact, we can find more theoretical evidences by comparing the results from previous separate works on QD and EA's runtime analysis, though not completely fair. For example, Bossek and Sudholt [2023] proved that for maximizing any monotone function over $\{0,1\}^n$, MAP-Elites that generates one initial solution and uses the number of 1-bits of a solution as the behavior descriptor can find an optimal solution in $O(n^2 \log n)$ expected time, while Lengler and Zou [2021] proved that $(\mu+1)$-EA with $\mu_0 \le \mu \le n$ (where μ_0 is some constant) needs superpolynomial time to optimize some monotone functions. In the future, we will theoretically verify the other benefit of QD algorithms that they can better create a map of high-performing solutions in the behavior space than separately searching for a high-performing solution in each cell of the behavior space, by analyzing the QD-Score metric, i.e., the sum of objective values across all the obtained solutions in the archive.

Many theoretical works are left to be done, e.g., comparing different QD frameworks (MAP-Elites vs. NSLC), and studying the influence of different components (e.g., parent selection and variation operators), which may inspire the design of better QD algorithms in practice. Another interesting work is to study the relationship between MAP-Elites and the corresponding multi-objective EA, GSEMO, as also indicated in [Bossek and Sudholt, 2023]. By treating the behavior descriptors as the extra objective functions to be optimized, GSEMO behaves somewhat similarly to MAP-Elites. The differences are: 1) MAP-Elites only compares solutions within the same cell (i.e., having the same behavior descriptor values) while GSEMO compares different behavior descriptor values based on the domination relationship; 2) MAP-Elites can control the granularity of the behavior space by setting the number of behavior descriptor values in a cell. The known results of MAP-Elites (including [Bossek and Sudholt, 2023] and ours) match that of GSEMO in [Neumann and Wegener, 2006; Friedrich *et al.*, 2010; Friedrich and Neumann, 2015; Qian *et al.*, 2019]. Thus, it would be interesting to show their performance gap on some problems, especially that MAP-Elites may overcome some difficulty of GSEMO due to its good diversity mechanism.

Acknowledgments

This work was supported by the National Science and Technology Major Project (2022ZD0116600) and National Science Foundation of China (62276124). Chao Qian is the corresponding author. The supplementary material is available at https://arxiv.org/abs/2401.10539.

References

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

[Bhatt *et al.*, 2022] V. Bhatt, B. Tjanaka, M. C. Fontaine, and S. Nikolaidis. Deep surrogate assisted generation of environments. In *NeurIPS*, New Orleans, LA, 2022.

[Bossek and Sudholt, 2023] J. Bossek and D. Sudholt. Runtime analysis of quality diversity algorithms. In *GECCO*, pages 1546–1554, Lisbon, Portugal, 2023.

[Chalumeau *et al.*, 2023] F. Chalumeau, R. Boige, B. Lim, V. Macé, M. Allard, A. Flajolet, A. Cully, and T. Pierrot. Neuroevolution is a competitive alternative to reinforcement learning for skill discovery. In *ICLR*, Kigali, Rwanda, 2023.

[Chvatal, 1979] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[Clune *et al.*, 2013] J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755):20122863, 2013.

[Colas *et al.*, 2020] C. Colas, V. Madhavan, J. Huizinga, and J. Clune. Scaling MAP-Elites to deep neuroevolution. In *GECCO*, pages 67–75, Cancún, Mexico, 2020.

[Conti *et al.*, 2018] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *NeurIPS*, pages 5032–5043, Montréal, Canada, 2018.

[Cully and Demiris, 2018] A. Cully and Y. Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2018.

[Cully *et al.*, 2015] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

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

[Doerr and Neumann, 2020] B. Doerr and F. Neumann. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, Cham, Switzerland, 2020.

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

[Doncieux *et al.*, 2019] S. Doncieux, A. Laflaquière, and A. Coninx. Novelty search: A theoretical perspective. In *GECCO*, pages 99–106, Prague, Czech Republic, 2019.

[Ecoffet *et al.*, 2021] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[Eysenbach *et al.*, 2018] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *ICLR*, Vancouver, Canada, 2018.

[Faldor *et al.*, 2023] M. Faldor, F. Chalumeau, M. Flageat, and A. Cully. MAP-Elites with descriptor-conditioned gradients and archive distillation into a single policy. In *GECCO*, pages 138–146, Lisbon, Portugal, 2023.

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

[Feng *et al.*, 2019] C. Feng, C. Qian, and K. Tang. Unsupervised feature selection by Pareto optimization. In *AAAI*, pages 3534–3541, Honolulu, HI, 2019.

[Fontaine *et al.*, 2021] M. C. Fontaine, R. Liu, A. Khalifa, J. Modi, J. Togelius, A. K. Hoover, and S. Nikolaidis. Illuminating Mario scenes in the latent space of a generative adversarial network. In *AAAI*, pages 5922–5930, Virtual, 2021.

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

[Friedrich *et al.*, 2010] 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.

[Grillotti and Cully, 2022] L. Grillotti and A. Cully. Unsupervised behavior discovery with quality-diversity optimization. *IEEE Transactions on Evolutionary Computation*, 26(6):1539–1552, 2022.

[Harshaw *et al.*, 2019] C. Harshaw, M. Feldman, J. Ward, and A. Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *ICML*, pages 2634–2643, Long Beach, CA, 2019.

[Jansen and Wegener, 2001] T. Jansen and I. Wegener. On the utility of populations in evolutionary algorithms. In *GECCO*, pages 1034–1041, San Francisco, CA, 2001.

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

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

[Kumar *et al.*, 2020] S. Kumar, A. Kumar, S. Levine, and C. Finn. One solution is not all you need: Few-shot extrapolation via structured MaxEnt RL. In *NeurIPS*, pages 8198–8210, Virtual, 2020.

[Lehman and Stanley, 2011a] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[Lehman and Stanley, 2011b] J. Lehman and K. O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *GECCO*, pages 211–218, Dublin, Ireland, 2011.

[Lengler and Zou, 2021] J. Lengler and X. Zou. Exponential slowdown for larger populations: The (μ+1)-EA on monotone functions. *Theoretical Computer Science*, 875:28–51, 2021.

[Lim *et al.*, 2022] B. Lim, M. C. Flageat, and A. Cully. Efficient exploration using model-based quality-diversity with gradients. *arXiv:2211.12610*, 2022.

[Lupu *et al.*, 2021] A. Lupu, H. Hu, and J. Foerster. Trajectory diversity for zero-shot coordination. In *ICML*, pages 7204–7213, Virtual, 2021.

[Mouret and Clune, 2015] J.-B. Mouret and J. Clune. Illuminating search spaces by mapping elites. *arXiv:1504.04909*, 2015.

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

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

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

[Nikfarjam *et al.*, 2022a] A. Nikfarjam, A. Neumann, and F. Neumann. On the use of quality diversity algorithms for the traveling thief problem. In *GECCO*, pages 260–268, Boston, MA, 2022.

[Nikfarjam *et al.*, 2022b] A. Nikfarjam, A. Viet Do, and F. Neumann. Analysis of quality diversity algorithms for the knapsack problem. In *PPSN*, pages 413–427, Dortmund, Germany, 2022.

[Nilsson and Cully, 2021] O. Nilsson and A. Cully. Policy gradient assisted MAP-Elites. In *GECCO*, page 866–875, Lille, France, 2021.

[Pugh *et al.*, 2016] J. K. Pugh, L. B. Soros, and K. O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers Robotics AI*, 3:40, 2016.

[Qian *et al.*, 2015a] C. Qian, Y. Yu, and Z.-H. Zhou. On constrained Boolean Pareto optimization. In *IJCAI*, pages 389–395, Buenos Aires, Argentina, 2015.

[Qian *et al.*, 2015b] C. Qian, Y. Yu, and Z.-H. Zhou. Subset selection by Pareto optimization. In *NeurIPS*, pages 1765–1773, Montreal, Canada, 2015.

[Qian *et al.*, 2016] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou. Parallel Pareto optimization for subset selection. In *IJCAI*, pages 1939–1945, New York, NY, 2016.

[Qian *et al.*, 2019] C. Qian, Y. Yu, K. Tang, X. Yao, and Z.-H. Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence*, 275:279–294, 2019.

[Salehi *et al.*, 2022] A. Salehi, A. Coninx, and S. Doncieux. Few-shot quality-diversity optimization. *IEEE Robotics and Automation Letters*, 7(2):4424–4431, 2022.

[Sfikas *et al.*, 2021] K. Sfikas, A. Liapis, and G. N. Yannakakis. Monte Carlo elites: Quality-diversity selection as a multi-armed bandit problem. In *GECCO*, pages 180–188, Lille, France, 2021.

[Storch, 2008] T. Storch. On the choice of the parent population size. *Evolutionary Computation*, 16(4):557–578, 2008.

[Strouse *et al.*, 2021] DJ Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett. Collaborating with humans without human data. In *NeurIPS*, pages 14502–14515, Virtual, 2021.

[Tjanaka *et al.*, 2022] B. Tjanaka, M. C. Fontaine, J. Togelius, and S. Nikolaidis. Approximating gradients for differentiable quality diversity in reinforcement learning. In *GECCO*, page 1102–1111, Boston, MA, 2022.

[Vassiliades *et al.*, 2018] V. Vassiliades, K. I. Chatzilygeroudis, and J.-B. Mouret. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation*, 22(4):623–630, 2018.

[Wang *et al.*, 2022] Y. Wang, K. Xue, and C. Qian. Evolutionary diversity optimization with clustering-based selection for reinforcement learning. In *ICLR*, Virtual, 2022.

[Wang *et al.*, 2023] R.-J. Wang, K. Xue, H. Shang, C. Qian, H. Fu, and Q. Fu. Multi-objective optimization-based selection for quality-diversity by non-surrounded-dominated sorting. In *IJCAI*, pages 4335–4343, Macao, SAR, China, 2023.

[Wiegand, 2024] R. P. Wiegand. Preliminary analysis of simple novelty search. *Evolutionary Computation*, pages 1–25, 2024.

[Witt, 2006] C. Witt. Runtime analysis of the (μ+1) EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14(1):65–86, 2006.

[Xue *et al.*, 2024] K. Xue, R.-J. Wang, P. Li, D. Li, J. Hao, and C. Qian. Sample-efficient quality-diversity by cooperative coevolution. In *ICLR*, Vienna, Austria, 2024.

[Zhang *et al.*, 2023] Y. Zhang, M. C. Fontaine, V. Bhatt, S. Nikolaidis, and J. Li. Multi-robot coordination and layout design for automated warehousing. In *IJCAI*, pages 5503–5511, Macao, SAR, China, 2023.

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