Multi-objective Optimization-based Selection for Quality-Diversity by Non-surrounded-dominated Sorting

Ren-Jian Wang[1], **Ke Xue**[1], **Haopu Shang**[1], **Chao Qian**[1*], **Haobo Fu**[2] and **Qiang Fu**[2]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[2]Tencent AI Lab, Shenzhen, China
{wangrj, xuek, shanghp, qianc}@lamda.nju.edu.cn, {haobofu, leonfu}@tencent.com

Abstract

Quality-Diversity (QD) algorithms, a subset of evolutionary algorithms, maintain an archive (i.e., a set of solutions) and simulate the natural evolution process through iterative selection and reproduction, with the goal of generating a set of high-quality and diverse solutions. Though having found many successful applications in reinforcement learning, QD algorithms often select the parent solutions uniformly at random, which lacks selection pressure and may limit the performance. Recent studies have treated each type of behavior of a solution as an objective, and selected the parent solutions based on Multi-objective Optimization (MO), which is a natural idea, but has not led to satisfactory performance as expected. This paper gives the reason for the first time, and then proposes a new MO-based selection method by non-surrounded-dominated sorting (NSS), which considers all possible directions of the behaviors, and thus can generate diverse solutions over the whole behavior space. By combining NSS with the most widespread QD algorithm, MAP-Elites, we perform experiments on synthetic functions and several complex tasks (i.e., QDGym, robotic arm, and Mario environment generation), showing that NSS achieves better performance than not only other MO-based selection methods but also state-of-the-art selection methods in QD.

1 Introduction

Generating a set of high-quality and diverse solutions is important in a wide variety of scenarios, such as robotics [Cully *et al.*, 2015; Allard *et al.*, 2022; Salehi *et al.*, 2022], combinatorial optimization [Do *et al.*, 2022; Nikfarjam *et al.*, 2022], multi-agent coordination [Lupu *et al.*, 2021; Xue *et al.*, 2022; Zhang *et al.*, 2023; Yu *et al.*, 2023], and reinforcement learning (RL) [Eysenbach *et al.*, 2018; Parker-Holder *et al.*, 2020; Chalumeau *et al.*, 2023]. For example, it is difficult for one single policy to adapt to a variety of situations when controlling a multi-foot robot, while maintaining a set of diverse policies (i.e., policies having different frequencies of using each foot) can improve robustness, e.g., enabling the robot to recover quickly from damage [Cully *et al.*, 2015].

Evolutionary algorithms (EAs) [Bäck, 1996; Zhou *et al.*, 2019] are general-purpose heuristic optimization algorithms that maintain a set of solutions, and simulate the natural evolution process by iterative reproduction and selection. Quality-Diversity (QD) algorithms [Cully *et al.*, 2015; Mouret and Clune, 2015; Cully and Demiris, 2018; Chatzilygeroudis *et al.*, 2021] are a specific type of EAs that aim to return a set of high-quality and diverse solutions in a single run. Given an objective function to be maximized and a behavior descriptor vector function, QD algorithms attempt to find a set of solutions that can cover the space of the behavior descriptor and have high objective values. Specifically, a QD algorithm maintains a set of solutions (called an archive), and iteratively performs the following process: selects a subset of parent solutions from the archive, then applies reproduction operators (e.g., crossover and mutation) to generate offspring solutions, and finally uses these offspring solutions to update the archive. The excellent performance of QD algorithms has been demonstrated in many RL tasks, such as exploration [Ecoffet *et al.*, 2021; Miao *et al.*, 2022], robust training [Kumar *et al.*, 2020; Yuan *et al.*, 2023a; Yuan *et al.*, 2023b], policy ensemble [Sheikh *et al.*, 2022], and environment generation [Bhatt *et al.*, 2022].

The parent selection strategy and offspring reproduction operator are critical to the performance of QD algorithms [Cully and Demiris, 2018]. Many recent works have been focusing on developing more efficient reproduction operators [Nilsson and Cully, 2021; Fontaine and Nikolaidis, 2021; Tjanaka *et al.*, 2022] to improve the sample efficiency, while using uniform random selection to select the parent solutions by default, which may, however, be inefficient due to the lack of selection pressure.

Multi-objective Optimization (MO) [Deb, 2011] considers the problems that optimize multiple conflicting objectives simultaneously, and the goal is to obtain a set of Pareto optimal solutions that can represent different trade-offs of the objectives. Thus, it is natural to view different types of behaviors as multiple objectives to optimize, and select the parent so-

*This work was supported by the National Science Foundation of China (62022039, 62276124), and the CCF-Tencent Open Research Fund (CCF-Tencent RAGR20220110). Chao Qian is the corresponding author.

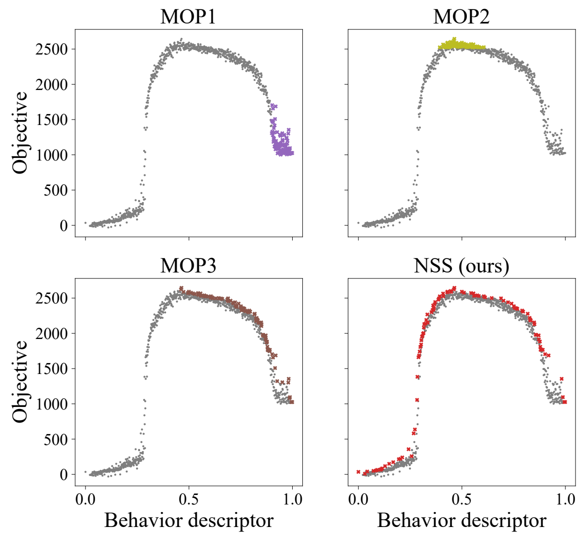

Figure 1: Solutions selected by different MO-based methods (i.e., the previous MOP1–3 and our proposed NSS) in the QD Hopper environment given the same archive. Selected solutions are marked with "$\times$" and corresponding colors. The x-axis represents the 1-dimensional behavior descriptor value (i.e., the fraction of time the single foot was touching the ground during an episode), and the y-axis represents the objective function value (i.e., the agent's forward speed) of a solution.

lutions based on MO before reproducing offspring solutions in QD [Shen *et al.*, 2020; Villin *et al.*, 2021]. However, MO-based selection strategies have not shown satisfactory performance as expected [Wang *et al.*, 2022].

In this paper, we point out the reason of the ineffectiveness of MO-based selection in QD for the first time. That is, optimizing multiple behavior descriptor functions simultaneously can obtain a set of diverse solutions with different preferences over behaviors, but actually only concentrates on a certain area of the behavior space (e.g., maximizing two behavior descriptor functions simultaneously will focus on the upper right part of the behavior space); while the goal of QD is to cover the whole behavior space. Figure 1 gives an example illustration of applying MO-based selection to the most popular QD algorithm, Multi-dimensional Archive of Phenotypic Elites (MAP-Elites, ME) [Cully *et al.*, 2015; Mouret and Clune, 2015]. ME discretizes the behavior space into M cells, each containing at most one solution, obtaining its archive. The goal of ME is to maximize the sum of the objective values of all solutions in the archive, called QD-Score. It can be clearly observed that the previous MO-based selection methods cannot cover the whole behavior space well, and thus can hardly help maximize the QD-Score.

To address the above issue of MO-based selection, we propose surrounded dominance, a new concept for solution comparison in MO. Traditionally, a solution Pareto dominates another one if it is not worse on all objectives and is better on at least one objective; thus, the behavior descriptor functions are optimized only in one direction. Now, surrounded dominance is a relationship between a solution and a solution set: a solution is surrounded dominated if for each possible direction of the behavior descriptor functions, there is a solution in the set which Pareto dominates it. Thus, by Non-Surrounded-dominated Sorting (NSS), the diverse solutions covering the behavior space well can be selected (as shown in the last subfigure of Figure 1), which can help improve the QD-Score efficiently. Note that the proposed NSS is a general selection method, which can be used in various QD algorithms, e.g., ME [Mouret and Clune, 2015], PGA-ME [Nilsson and Cully, 2021], and OG-ME [Fontaine and Nikolaidis, 2021].

We conduct experiments on synthetic functions and popular benchmarks (i.e., QDGym [Nilsson and Cully, 2021; Flageat *et al.*, 2023], robotic arm [Cully *et al.*, 2015; Fontaine and Nikolaidis, 2021], and Mario environment generation [Bhatt *et al.*, 2022]) to examine the performance of NSS. As expected, the results show that NSS achieves better performance (i.e., QD-Score) than the other MO-based selection methods [Shen *et al.*, 2020; Villin *et al.*, 2021]. Furthermore, it can perform better, even compared with the state-of-the-art selection methods [Wang *et al.*, 2022].

2 Background

2.1 Quality Diversity

QD algorithms aim to find a diverse set of high-quality solutions of a problem [Cully and Demiris, 2018; Chatzilygeroudis *et al.*, 2021]. Let $\mathcal{X}$ denote the solution space, and $\mathcal{S} \subseteq \mathbb{R}^k$ denote the k-dimensional descriptor space. Given an objective (quality) function $f : \mathcal{X} \to \mathbb{R}$ to be maximized and a behavior descriptor function $\boldsymbol{m} : \mathcal{X} \to \mathcal{S}$, the goal of QD algorithms is to find solutions that span the k-dimensional descriptor space $\mathcal{S}$ while maximizing the objective function f.

Take the most well-known QD algorithm, ME [Cully *et al.*, 2015; Mouret and Clune, 2015], as an example. It maintains an archive by discretizing the descriptor space $\mathcal{S}$ into M cells $\{\mathcal{S}_i\}_{i=1}^M$ and storing at most one solution in each cell. ME tries to fill in the cells with as high-quality solutions as possible. That is, the goal of ME can be formalized as maximizing the QD-Score:

$$\sum\nolimits_{i=1}^{M} f(\boldsymbol{x}_i), \tag{1}$$

where $\boldsymbol{x}_i$ denotes the solution contained by the cell $\mathcal{S}_i$, i.e., $\boldsymbol{m}(\boldsymbol{x}_i) \in \mathcal{S}_i$. Note that if a cell $\mathcal{S}_i$ does not have a solution $\boldsymbol{x}_i$, then $f(\boldsymbol{x}_i)$ is defined as 0. Without loss of generality, the objective value $f(\cdot)$ is assumed (or converted) to be non-negative, to prevent solutions from decreasing the QD-Score.

The main procedure of QD algorithms is to iteratively select parent solutions from the archive, generate offspring solutions by reproduction operators, and update the archive. The selection method, a key component of QD algorithms [Cully and Demiris, 2018], aims to address the following question [Chatzilygeroudis *et al.*, 2021]: *Given the current archive, how do we select appropriate parent solutions to generate new offspring solutions?* Uniform random selection, i.e., selecting parent solutions from the archive uniformly at random, is one of the simplest selection methods and has been widely used in QD algorithms such as [Cully *et al.*, 2015; Nilsson and Cully, 2021; Fontaine and Nikolaidis, 2021; Tjanaka *et al.*, 2022]. Recently, Wang *et al.* [2022] proposed a clustering-based selection method EDO-CS, which divides the archive into several clusters and then selects good parent solutions from each cluster. They compared different selection methods on various continuous control tasks, showing the state-of-the-art performance of EDO-CS and also demonstrating the importance of an efficient selection method.

2.2 Multi-objective Optimization-based Selection

Another branch of selection methods for QD are based on Multi-objective Optimization (MO), which tries to maximize multiple objective functions $f_1, \ldots, f_k : \mathcal{X} \to \mathbb{R}$ simultaneously, i.e.,

$$\max_{\boldsymbol{x} \in \mathcal{X}} \; (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})) . \tag{2}$$

The objective function vector $(f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$ is also represented as $\boldsymbol{f}(\boldsymbol{x})$ for convenience. Since the objectives are usually conflicting, two solutions may be incomparable, and the solution comparison is usually based on the Pareto dominance relationship [Deb, 2011], as shown in Definition 1. This also implies that there is no single optimal solution in MO, but rather a set of Pareto optimal solutions.

Definition 1 (Pareto Dominance)**.** *A solution $\boldsymbol{x}$ Pareto dominates $\boldsymbol{x}'$ (denoted as $\boldsymbol{x} \succ \boldsymbol{x}'$) if $\forall i : f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}')$ and $\exists i : f_i(\boldsymbol{x}) > f_i(\boldsymbol{x}')$. A solution $\boldsymbol{x}^*$ is Pareto optimal if $\nexists \boldsymbol{x} \in \mathcal{X}$ such that $\boldsymbol{x} \succ \boldsymbol{x}^*$. The set of all Pareto optimal solutions is called Pareto Set (PS). The set of the corresponding objective vectors of PS, i.e., $\{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in PS\}$, is called Pareto Front (PF).*

As the size of PF can be exponentially large, the goal of MO is often to find a set of Pareto optimal solutions, which can approximate the PF well. Furthermore, each Pareto optimal solution represents a unique trade-off between the objectives. Thus, the obtained set of solutions can be seen as *diverse* from the perspective of multiple objectives, and MO has been naturally used to select diverse parent solutions in QD [Shen *et al.*, 2020; Villin *et al.*, 2021]. There have been three types of MO formulation for parent selection in QD. MOP1 [Villin *et al.*, 2021] treats each behavior descriptor function $m_i : \mathcal{X} \to \mathbb{R}$ as an objective, i.e.,

$$\max_{\boldsymbol{x} \in \mathcal{X}} \; (m_1(\boldsymbol{x}), \ldots, m_k(\boldsymbol{x})) , \tag{3}$$

which promotes diversity only. MOP2 [Shen *et al.*, 2020; Villin *et al.*, 2021] considers quality into the MO formulation:

$$\max_{\boldsymbol{x} \in \mathcal{X}} \; ([f(\boldsymbol{x}), m_1(\boldsymbol{x})], \ldots, [f(\boldsymbol{x}), m_k(\boldsymbol{x})]) , \tag{4}$$

where for each objective $[f(\boldsymbol{x}), m_i(\boldsymbol{x})]$, $f(\boldsymbol{x})$ is prioritized over $m_i(\boldsymbol{x})$, i.e., $\boldsymbol{x}$ is better than $\boldsymbol{x}'$ if $f(\boldsymbol{x}) > f(\boldsymbol{x}')$, or $f(\boldsymbol{x}) = f(\boldsymbol{x}') \wedge m_i(\boldsymbol{x}) > m_i(\boldsymbol{x}')$. MOP3 [Villin *et al.*, 2021] also considers quality, but treats the quality function f as another objective directly, i.e.,

$$\max_{\boldsymbol{x} \in \mathcal{X}} \; (f(\boldsymbol{x}), m_1(\boldsymbol{x}), \ldots, m_k(\boldsymbol{x})) . \tag{5}$$

However, these MO-based selection methods do not lead to satisfactory performance of QD. This is because maximizing multiple behavior descriptor functions simultaneously actually only focuses on a certain area of the behavior space, while the goal of QD is to cover the whole behavior space. This has been observed in Figure 1, when MOP1–3 are applied to ME in the QD Hopper environment.

3 Method

In this section, we first introduce a new concept of surrounded dominance for solution comparison in Section 3.1, which considers all the possible directions of the behavior space. Based on this, we propose the Non-Surrounded-dominated Sorting (NSS) for selecting parent solutions in QD in Section 3.2, and analyze its property in Section 3.3. Finally, we use NSS as the selection method of ME, resulting in the algorithm NSS-ME in Section 3.4.

3.1 Surrounded Dominance

As discussed in Section 2.2, the classical MO-based selection methods based on Pareto dominance prefer the solutions in a certain direction of the behavior space (i.e., maximizing all the behavior descriptor functions simultaneously). This issue makes them hard to improve the performance of QD algorithms efficiently, whose goal is to obtain a set of high-quality solutions covering the whole behavior space. To address this issue, we propose a new concept called surrounded dominance, which considers all possible directions of the behavior space rather than a certain direction.

Definition 2 (Surrounded Dominance)**.** *For a solution $\boldsymbol{x}$ and a solution set $\mathcal{A}$, $\boldsymbol{x}$ is surrounded dominated by $\mathcal{A}$ if for each $\boldsymbol{d} \in \mathcal{D} = \{-1, 1\}^k$, there exists a solution $\boldsymbol{x}' \in \mathcal{A}$ such that $\boldsymbol{x}' \succ \boldsymbol{x}$ with respect to the objective vector $\boldsymbol{d} \odot \boldsymbol{m}(\cdot)$, where $\odot$ denotes the element-wise product of two vectors.*

Different from classical Pareto dominance, surrounded dominance is a relationship between a solution and a solution set. It uses a vector in $\{-1, 1\}^k$ to represent one direction (where k denotes the number of behavior descriptor functions), and the behavior descriptor vector $\boldsymbol{m}(\cdot) = (m_1(\cdot), \ldots, m_k(\cdot))$ is multiplied by this vector for comparison in this direction. A solution $\boldsymbol{x}$ is surrounded dominated by a solution set $\mathcal{A}$ if for each direction $\boldsymbol{d} \in \{-1, 1\}^k$, there is a solution $\boldsymbol{x}'$ in $\mathcal{A}$ better than $\boldsymbol{x}$, i.e., $\boldsymbol{x}' \succ \boldsymbol{x}$ with respect to $\boldsymbol{d} \odot \boldsymbol{m}(\cdot)$. That is, the behavior descriptors are considered in all possible directions for comparison.

However, surrounded dominance only considers the behavior descriptor functions $m_1(\cdot), \ldots, m_k(\cdot)$ (i.e., diversity) but ignores the objective function $f(\cdot)$ (i.e., quality), making it only have the ability of exploring the behavior space but hardly improve the quality of the solutions or spread the high-quality solutions to other areas. To address this issue, we further propose surrounded dominance with quality.

Definition 3 (Surrounded Dominance with Quality)**.** *For a solution $\boldsymbol{x}$ and a solution set $\mathcal{A}$, $\boldsymbol{x}$ is surrounded dominated by $\mathcal{A}$ if for each $\boldsymbol{d} \in \mathcal{D} = \{-1, 1\}^k$, there exists a solution $\boldsymbol{x}' \in \mathcal{A}$ such that $f(\boldsymbol{x}') > f(\boldsymbol{x})$ and $\boldsymbol{x}' \succ \boldsymbol{x}$ with respect to the objective vector $\boldsymbol{d} \odot \boldsymbol{m}(\cdot)$, where $\odot$ denotes the element-wise product of two vectors.*

That is, a solution $\boldsymbol{x}$ non-surrounded-dominated by a solution set $\mathcal{A}$ implies that there exists one direction $\boldsymbol{d} \in \{-1, 1\}^k$ such that no solution $\boldsymbol{x}' \in \mathcal{A}$ satisfies that $f(\boldsymbol{x}') > f(\boldsymbol{x})$ and $\boldsymbol{x}' \succ \boldsymbol{x}$ with respect to $\boldsymbol{d} \odot \boldsymbol{m}(\cdot)$. Thus, by selecting non-surrounded-dominated solutions for reproduction, QD algorithms can either explore new area of the behavior space or improve the quality of explored areas. In the absence of ambiguity, surrounded dominance is defaulted to the version with quality for simplicity. An example for surround dominance is provided in Appendix A.

3.2 Non-surrounded-dominated Sorting

Based on surrounded dominance, we propose the NSS method to sort the solutions in an archive $\mathcal{A}$. Specifically, NSS divides the solutions in $\mathcal{A}$ into several hierarchical non-surrounded-dominated fronts $\{\mathcal{P}_i\}_{i\in\mathcal{I}}$, where $\mathcal{P}_i$ denotes the i-th front, and $|\mathcal{I}|$ is the total number of fronts. The first front $\mathcal{P}_1$ contains the solutions that are not surrounded dominated by $\mathcal{A}$. After determining $\{\mathcal{P}_j\}_{j\in\{1,2,\dots,i\}}$, a solution in $\mathcal{A}\setminus\{\mathcal{P}_j\}_{j\in\{1,2,\dots,i\}}$ (i.e., the remaining solution set after removing the solutions in $\{\mathcal{P}_j\}_{j\in\{1,2,\dots,i\}}$ from the archive $\mathcal{A}$) is added into $\mathcal{P}_{i+1}$ if it is not surrounded dominated by $\mathcal{A}\setminus\{\mathcal{P}_j\}_{j\in\{1,2,\dots,i\}}$. According to the definition of surrounded dominance, the solutions in $\mathcal{P}_1$ have less-explored or low-quality neighbor area in the behavior space, implying that the offspring solutions generated from the solutions in $\mathcal{P}_1$ are more likely to explore the areas of the behavior space that are not reached and spread the high-quality solutions to other areas. Therefore, it is natural to select the solutions from the top-ranked fronts as the parent solutions during the reproduction process of QD algorithms to improve their performance. An example illustration in Figure 1 has clearly shown that NSS can select more diverse solutions in the behavior space than previous MO-based selection methods.

The detailed procedure of NSS is presented in Algorithm 1. For each solution $\boldsymbol{x}$ in the archive $\mathcal{A}$, NSS first initializes a dominance counter vector $\boldsymbol{c}(\boldsymbol{x})\in\mathbb{R}^{|\mathcal{D}|}$, where $\mathcal{D}=\{-1,1\}^k$ denotes the direction vector set, k is the number of behavior descriptor functions, and the i-th dimension $c_i(\boldsymbol{x})$ is the number of solutions $\boldsymbol{x}'$ satisfying that $f(\boldsymbol{x}')>f(\boldsymbol{x})$ and $\boldsymbol{x}'\succ\boldsymbol{x}$ with respect to the i-th direction $\boldsymbol{d}_i\odot\boldsymbol{m}(\cdot)$ of the behavior space. Note that $\boldsymbol{d}_i$ denotes the i-th direction vector in $\mathcal{D}$. Accordingly, NSS initializes $|\mathcal{D}|$ dominating sets $\{Q_i(\boldsymbol{x})\}_{i=1}^{|\mathcal{D}|}$ for each solution $\boldsymbol{x}$, where $Q_i(\boldsymbol{x})$ contains all the solutions $\boldsymbol{x}'$ such that $f(\boldsymbol{x})>f(\boldsymbol{x}')$ and $\boldsymbol{x}\succ\boldsymbol{x}'$ with respect to $\boldsymbol{d}_i\odot\boldsymbol{m}(\cdot)$. If the dominance counter vector $\boldsymbol{c}(\boldsymbol{x})$ of a solution $\boldsymbol{x}$ contains 0 elements, $\boldsymbol{x}$ is not surrounded dominated by the archive $\mathcal{A}$ according to Definition 3, and thus will be put into $\mathcal{P}_1$ in line 2. After determining $\mathcal{P}_1$, the dominance counter vectors of the remaining solutions in $\mathcal{A}\setminus\mathcal{P}_1$ will be updated by ignoring those solutions in $\mathcal{P}_1$, and the solutions with at least one 0 element in their dominance counter vectors will be put into the second front $\mathcal{P}_2$. This process will be repeated until all the solutions have been put into the corresponding non-surrounded-dominated fronts, as shown in lines 4–15. In particular, after determining $\mathcal{P}_i$, NSS checks the solutions in the dominating set $Q_j(\boldsymbol{x})$, for each solution $\boldsymbol{x}\in\mathcal{P}_i$ and each direction $\boldsymbol{d}_j\in\mathcal{D}$ in line 6. If the dominance counter vector $\boldsymbol{c}(\boldsymbol{x}')$ of a solution $\boldsymbol{x}'\in Q_j(\boldsymbol{x})$ does not have 0 elements, i.e., $\boldsymbol{x}'$ has not been put into the fronts, then its dominance counter $c_j(\boldsymbol{x}')$ in the corresponding direction $\boldsymbol{d}_j$ will be reduced by 1 in line 8. If $c_j(\boldsymbol{x}')$ is reduced to 0, implying that $\boldsymbol{x}'$ must be not surrounded dominated by the remaining solution set $\mathcal{A}\setminus\{\mathcal{P}_j\}_{j\in\{1,2,\dots,i\}}$, then it will be put into the next front $\mathcal{P}_{i+1}$ in line 10.

Algorithm 1 Non-surrounded-dominated Sorting

```
Input: archive $\mathcal{A}$
Output: hierarchical non-surrounded-dominated fronts $\{\mathcal{P}_1, \mathcal{P}_2, \dots\}$ of the archive $\mathcal{A}$
1:  For each solution $\boldsymbol{x}\in\mathcal{A}$, initialize its dominance counter vector $\boldsymbol{c}(\boldsymbol{x})$ and dominating sets $\{Q_i(\boldsymbol{x})\}_{i=1}^{|\mathcal{D}|}$;
2:  Put all the solutions whose dominance counter vector has at least one 0 element into $\mathcal{P}_1$;
3:  Let $i=1$;
4:  while $\mathcal{P}_i\neq\emptyset$ do
5:      Let $\mathcal{P}_{i+1}\leftarrow\emptyset$;
6:      for all $\boldsymbol{x}\in\mathcal{P}_i$, $j\in\{1,2,\dots,|\mathcal{D}|\}$, $\boldsymbol{x}'\in Q_j(\boldsymbol{x})$ do
7:          if $\boldsymbol{c}(\boldsymbol{x}')$ does not have 0 elements then
8:              $c_j(\boldsymbol{x}')\leftarrow c_j(\boldsymbol{x}')-1$;
9:              if $c_j(\boldsymbol{x}')=0$ then
10:                 $\mathcal{P}_{i+1}\leftarrow\mathcal{P}_{i+1}\cup\{\boldsymbol{x}'\}$
11:             end if
12:         end if
13:     end for
14:     $i\leftarrow i+1$
15: end while
16: return $\{\mathcal{P}_j\}_{j\in\{1,2,\dots,i-1\}}$
```

3.3 Property of NSS

Let $\mathcal{A}_{\text{opt}}$ denote the optimal archive, which consists of the highest-quality solution for each point in the behavior space. Next, we prove in Theorem 1 that a surrounded dominated solution will not appear in $\mathcal{A}_{\text{opt}}$, implying that considering only non-surrounded-dominated solutions by the NSS-based selection will not miss good solutions. The proof utilizes a behavior-quality function $g_{\mathcal{A}_{\text{opt}}}:\mathcal{S}\to\mathbb{R}$ as follows:

$$\forall\boldsymbol{b}\in\mathcal{S},\ g_{\mathcal{A}_{\text{opt}}}(\boldsymbol{b})=f(\boldsymbol{x_b}),\tag{6}$$

where $\boldsymbol{x_b}$ is the solution having behavior $\boldsymbol{b}$ in $\mathcal{A}_{\text{opt}}$, i.e., $\boldsymbol{x_b}\in\mathcal{A}_{\text{opt}}$ and $\boldsymbol{m}(\boldsymbol{x_b})=\boldsymbol{b}$. It also relies on an assumption that $g_{\mathcal{A}_{\text{opt}}}$ is quasi-concave, which intuitively means that the further away from the optimal behavior (i.e., the behavior with the highest quality), the worse the quality of the behavior. This assumption often holds in QD problems. For example, in QD Walker [Nilsson and Cully, 2021], the goal is to make the agent walk fast, where the quality is measured by the agent's forward speed, and the two-dimensional behavior vector function is described by the fraction of time each foot was touching the ground during an episode. Obviously, the agent can walk fast only when the fractions of time are appropriate; the greater the fractions deviate from the appropriate values, the slower the agent walks.

Theorem 1. *A surrounded dominated solution will not appear in the optimal archive* $\mathcal{A}_{\text{opt}}$.

Proof. Assume that a solution $\boldsymbol{x}$ in the optimal archive $\mathcal{A}_{\text{opt}}$ is surrounded dominated by an archive $\mathcal{A}$. According to the definition of surrounded dominance, i.e., Definition 3, we have for each direction $\boldsymbol{d}_i\in\mathcal{D}$, there exists a solution $\boldsymbol{x}_i$ such that $f(\boldsymbol{x}_i)>f(\boldsymbol{x})$ and $\boldsymbol{x}_i\succ\boldsymbol{x}$ with respect to the objective vector $\boldsymbol{d}_i\odot\boldsymbol{m}(\cdot)$. Let $\mathcal{B}=\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_{|\mathcal{D}|}\}$. It is obvious that $\boldsymbol{m}(\boldsymbol{x})$ is in the convex hull of the set $\{\boldsymbol{m}(\boldsymbol{x}')\mid\boldsymbol{x}'\in\mathcal{B}\}$.

Algorithm 2 NSS-ME

Parameter: number T of total generations, number N of selected solutions in each generation
Output: archive $\mathcal{A}$

```
1: Let $\mathcal{A} \leftarrow \emptyset, t \leftarrow 1$;
2: while $t \leq T$ do
3:   if $t = 1$ then
4:     $\mathcal{B}_o^{(t)} \leftarrow \mathrm{Randomly\_Generate}(N)$
5:   else
6:     $\mathcal{B}_p^{(t)} \leftarrow \mathrm{NSS\_Select}(\mathcal{A}, N)$;
7:     $\mathcal{B}_o^{(t)} \leftarrow \mathrm{Reproduction}(\mathcal{B}_p^{(t)})$
8:   end if
9:   $\mathrm{Evaluate}(\mathcal{B}_o^{(t)})$;
10:  $\mathrm{Update\_Archive}(\mathcal{A}, \mathcal{B}_o^{(t)})$;
11:  $t \leftarrow t + 1$
12: end while
13: return $\mathcal{A}$
```

Let $f_{\mathcal{B}\min} = \min_{\boldsymbol{x}' \in \mathcal{B}} f(\boldsymbol{x}')$. By the quasi-concave assumption of the behavior-quality function $g_{\mathcal{A}_{\mathrm{opt}}}$ in Eq. (6), the upper-level behavior set $\mathcal{S}_{f_{\mathcal{B}\min}}$ of $g_{\mathcal{A}_{\mathrm{opt}}}$ with respect to the threshold $f_{\mathcal{B}\min}$ is a convex set. Since $f(\boldsymbol{x}') \geq f_{\mathcal{B}\min}$ for any $\boldsymbol{x}' \in \mathcal{B}$, the convex hull of $\{\boldsymbol{m}(\boldsymbol{x}') \mid \boldsymbol{x}' \in \mathcal{B}\}$ must be a subset of $\mathcal{S}_{f_{\mathcal{B}\min}}$. Thus, we have $\boldsymbol{m}(\boldsymbol{x}) \in \mathcal{S}_{f_{\mathcal{B}\min}}$, implying $f(\boldsymbol{x}) \geq f_{\mathcal{B}\min} = \min_{\boldsymbol{x}' \in \mathcal{B}} f(\boldsymbol{x}')$, contradicting $\forall \boldsymbol{x}' \in \mathcal{B} : f(\boldsymbol{x}) < f(\boldsymbol{x}')$. Thus, the theorem holds. □

3.4 NSS-based QD Algorithms

Finally, we apply NSS-based selection to the most popular framework ME of QD algorithms, resulting in NSS-ME, as presented in Algorithm 2. We will use NSS-ME to examine the effectiveness of NSS in the experiments.

At the beginning, the archive $\mathcal{A}$ is created as an empty set in line 1, and the N offspring solutions $\mathcal{B}_o^{(1)}$ are randomly generated in line 4. After that, in each generation t (where $t > 1$), NSS-ME first selects N parent solutions $\mathcal{B}_p^{(t)}$ from the archive $\mathcal{A}$ by $\mathrm{NSS\_Select}(\mathcal{A}, N)$ in line 6, which selects N solutions from the top-ranked non-surrounded-dominated fronts of $\mathcal{A}$ obtained by NSS in Algorithm 1. Let $\{\mathcal{P}_i\}_{i \in \mathcal{I}}$ denote the non-surrounded-dominated fronts of $\mathcal{A}$, and let $j^* = \max\{j \geq 0 \mid \sum_{i=1}^{j} |\mathcal{P}_i| \leq N\}$. The N parent solutions $\mathcal{B}_p^{(t)}$ consists of $\{\mathcal{P}_i\}_{i \in \{1,2,\ldots,j^*\}}$ and $N - \sum_{i=1}^{j^*} |\mathcal{P}_i|$ solutions randomly selected from $\mathcal{P}_{j^*+1}$. The offspring solutions $\mathcal{B}_o^{(t)}$ are then generated via reproduction operators in line 7, such as Gaussian mutation in ME [Mouret and Clune, 2015], objective gradient in OG-ME [Fontaine and Nikolaidis, 2021], and policy gradient in PGA-ME [Nilsson and Cully, 2021; Flageat *et al.*, 2023]. After evaluating $\mathcal{B}_o^{(t)}$, NSS-ME updates the archive $\mathcal{A}$ based on the rules of ME methods [Mouret and Clune, 2015; Nilsson and Cully, 2021]. That is, each solution $\boldsymbol{x} \in \mathcal{B}_o^{(t)}$ is placed in its corresponding cell in the behavior space according to its behavior $\boldsymbol{m}(\boldsymbol{x})$. If the cell is empty, $\boldsymbol{x}$ is kept directly; otherwise, the one with a higher quality between $\boldsymbol{x}$ and the solution occupying the cell is kept.

4 Experiments

To examine the performance of NSS, we conduct experiments on synthetic functions and several complex tasks, including QDGym, robotic arm, and Mario environment generation. We consider the following three main metrics for evaluation.

- **QD-Score**: The total sum of objective (quality) values across all solutions in the archive, as defined in Eq. (1). It reflects both the quality and diversity of the solutions, and is the most important metric to evaluate a QD algorithm [Pugh *et al.*, 2016; Cully and Demiris, 2018].
- **Coverage**: The total number of solutions in the archive. It can measure the exploration ability of a QD algorithm.
- **Best Performance**: The largest objective value of solutions in the archive. It can measure the exploitation ability of a QD algorithm.

For each task, we will compare NSS with other parent selection methods under the same implementation of ME for fair comparison. We run each method multiple times independently, and report the average results. The detailed settings of experiments are provided in Appendix B. Our code is available at https://github.com/lamda-bbo/NSS.

4.1 Synthetic Functions

First, we conduct experiments on a simple synthetic function to show the advantage of NSS over other MO-based selection methods intuitively. We define a 2-dimensional behavior descriptor vector function adopted from [Fontaine and Nikolaidis, 2021]: the first and second dimensions are the sum of the first and second half of the elements of a real-valued vector solution, respectively. The objective function is defined as the sum of the distance between each element of the solution and the average of the half the element belongs to. We compare NSS with random selection and previous MO-based methods, including MOP1, MOP2, and MOP3 in Eqs. (3)–(5) [Shen *et al.*, 2020; Villin *et al.*, 2021]. All of these selection methods use ME [Mouret and Clune, 2015] as the framework and all the other components are set to the same.

As shown in Figure 2, NSS achieves the highest QD-Score and fills the largest number of cells in the behavior descriptor space, demonstrating its superior overall performance and exploration ability. Although MOP2 and MOP3 have a little higher best performance than NSS, they cannot cover the behavior space well, and their QD-Scores are extremely low. MOP1 performs badly in all three metrics. These previous MO-based selection methods are even worse than random selection on QD-Score and archive coverage. Figure 3 also visualizes the archives obtained by different MO-based methods. We can clearly observe that the solutions obtained by MOP1–3 are concentrated in the upper right corner of the behavior space, while NSS can cover the behavior space well.

4.2 Complex Tasks

QDGym. It is a popular benchmark to evaluate QD algorithms [Nilsson and Cully, 2021; Tjanaka *et al.*, 2022; Flageat *et al.*, 2023]. We conduct experiments on four different environments, i.e., QD Hopper, Walker, HalfCheetah, and Ant. These tasks aim to generate a set of policies that move

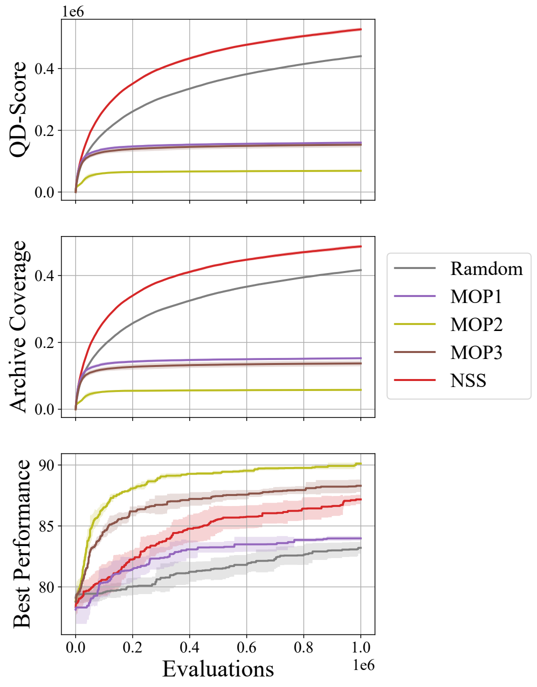

Figure 2: Plots of QD-Score, archive coverage and best performance for random, NSS and other MO-based selection methods on the synthetic function. The x-axis is the number of solutions evaluated.

forward as fast as possible and are diverse in the frequency of feet use. Thus, the objective function is determined by the agent's forward speed, and the behavior descriptor functions are defined as the fraction of time each foot was touching the ground during an episode. The ME framework here uses the implementation in [Mouret and Clune, 2015]. In addition, we also conduct the experiments with PGA-ME [Nilsson and Cully, 2021; Flageat *et al.*, 2023], achieving similar results which are shown in Appendix C.1 due to space limitation.

Robotic Arm. It aims to find a set of joint angle solutions for a robotic arm that allows the end effector positions to cover its reachable space [Cully *et al.*, 2015; Fontaine and Nikolaidis, 2021]. The objective function is defined as the variance of the joint angles, and the behavior descriptor functions are defined as the positions of the end effector. All the parent selection methods on this task use OG-ME [Fontaine and Nikolaidis, 2021] as the framework.

Mario Environment Generation. The QD algorithms can be used to generate a set of diverse environments [Fontaine *et al.*, 2021; Bhatt *et al.*, 2022]. We conduct experiments on Mario environment generation, which aims to generate a set of Mario environments with diverse properties. The generated environments are evaluated by simulating a pre-trained agent in the environments provided in [Bhatt *et al.*, 2022]. The objective function is defined as the completion rate. The two behavior descriptor functions are defined as the number of tiles of a certain type that are in the upper half of the 2D grid, and the number of jumps of the agent, respectively. We use the implementation of ME in [Mouret and Clune, 2015].

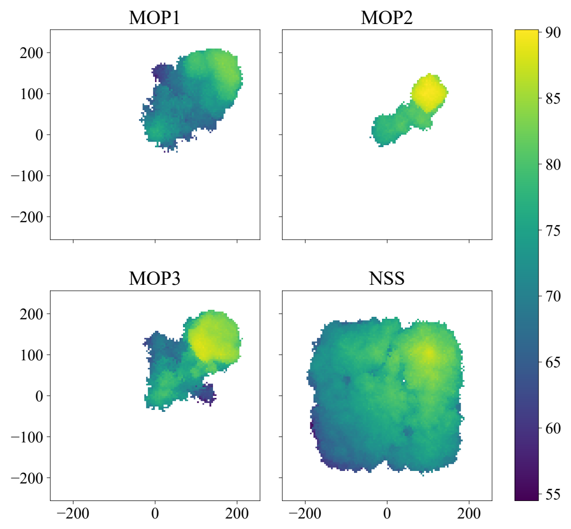

Figure 3: Visualization of archives post-trained on the synthetic function. The behavior space is divided into a 2-dimensional grid. Cells are left blank when no solution with the corresponding behavior descriptor exists in the archive, and otherwise colored as the objective values of the corresponding solutions.

Besides random selection and MOP1–3 compared in Section 4.1, we run three more parent selection methods here. NSLC [Lehman and Stanley, 2011] considers both the novelty score and local quality score (i.e., the number of neighbors that a solution outperforms), and selects parent solutions from the corresponding Pareto front obtained by NSGA-II [Deb *et al.*, 2002]. Curiosity-based selection [Cully and Demiris, 2018] calculates the curiosity score based on the history information (i.e., increases the score if the offspring solution improves the QD-Score; otherwise, decreases it), and selects the solutions with higher curiosity scores. EDO-CS [Wang *et al.*, 2022] divides the archive into several clusters and then selects good parent solutions from each cluster. It has been shown that EDO-CS achieves state-of-the-art performance in many tasks [Wang *et al.*, 2022]. For these three methods, we use their recommended hyperparameters.

The results are shown in Table 1. Instead of simply showing the QD-Score of the returned archive, we use QD-Score AUC [Tjanaka *et al.*, 2022],

$$\sum\nolimits_{t=1}^{T} (\lambda \times \text{QD-Score at generation } t)\,, \qquad (7)$$

where T is the total number of generations, and λ is the number of evaluated solutions per generation. That is, the QD-Score AUC measures the area under the QD-Score curve, which quantifies the optimization efficiency of a QD algorithm. It can be observed from Table 1 that NSS has the best average rank, EDO-CS is the runner-up, and the previous MO-based selection methods (i.e., MOP1–3) performs the worst (even worse than random selection). We also perform the Wilcoxon rank-sum test with significance level 0.05, and the row '$+/-/\approx$' shows the clear advantage of NSS over other methods. For the running time, NSS-based selection costs 0.0618s per generation on QD HalfCheetah, which is

Environment	Random	MOP1	MOP2	MOP3	NSLC	Curiosity	EDO-CS	NSS
QD Hopper	1.25 –	0.88 –	1.26 –	1.17 –	0.78 –	1.53 –	1.44 –	**1.69**
QD Walker	0.34 –	0.34 –	0.37 –	0.35 –	0.29 –	0.36 ≈	0.38 ≈	**0.40**
QD HalfCheetah	3.72 –	2.94 –	3.17 –	3.08 –	3.48 –	3.84 ≈	**3.94** ≈	3.90
QD Ant	1.02 ≈	0.77 –	0.88 –	0.80 –	0.90 –	1.05 ≈	**1.09** ≈	1.04
Arm	18.09 –	17.26 –	9.89 –	20.17 –	**21.77** +	18.30 –	18.59 –	20.52
Mario	98.45 –	41.05 –	38.29 –	40.54 –	112.72 –	**138.65** ≈	108.55 –	134.18
+/–/≈	0/5/1	0/6/0	0/6/0	0/6/0	1/5/0	0/2/4	0/3/3	
Average Rank	5.00	7.00	5.83	5.83	5.00	2.83	2.50	**1.83**

Table 1: QD-Score AUC (first four rows $\times 10^{12}$, last two rows $\times 10^{6}$) of different methods on six complex tasks. The symbols '+', '–', and '≈' indicate that the result is significantly superior to, inferior to, and almost equivalent to NSS, respectively, according to the Wilcoxon rank-sum test with significance level 0.05.

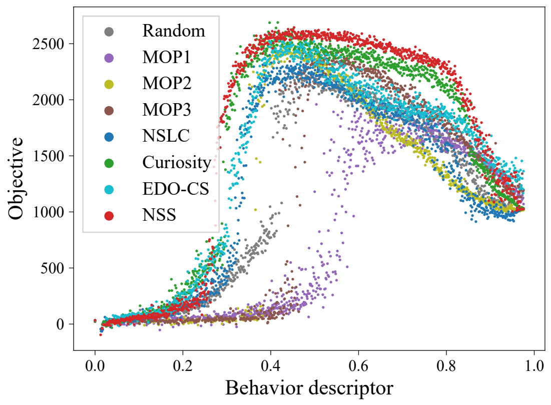

Figure 4: Visualization of archives post-trained on QD Hopper. The x-axis represents the 1-dimensional behavior descriptor value, and the y-axis represents the objective function value of a solution.

much faster than 0.2918s of EDO-CS, which requires clustering for selection. The complete results about running time are shown in Appendix C.2. Figure 4 visualizes the archives obtained by different methods on QD Hopper, which clearly demonstrates the superiority of NSS. Compared with other methods, NSS can cover a large area of the behavior space, and generate more high-quality solutions. More visualization results are shown in Appendix C.3.

4.3 Additional Results

In our experiments, we have used surrounded dominance with quality in Definition 3 when implementing NSS-based selection. Here, we conduct ablation study to compare it with the version without quality (i.e., using surrounded dominance in Definition 2). Figure 5 shows that NSS archives higher QD-Score and best performance, validating the importance of considering the quality in surrounded dominance.

Furthermore, we analyze the influence of the number N of selected solutions in each generation. Figure 6 shows that a too small or too large N may lead to relatively worse performance, but the performance is overall not very sensitive to N. Note that N is a hyperparameter that comes with the ME framework, while NSS is a selection method *without additional hyperparameters*. On the contrary, other competitive

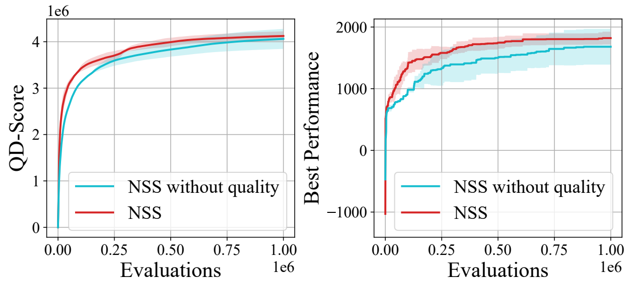

Figure 5: QD-Score and best performance for NSS with and without quality on QD Hopper.

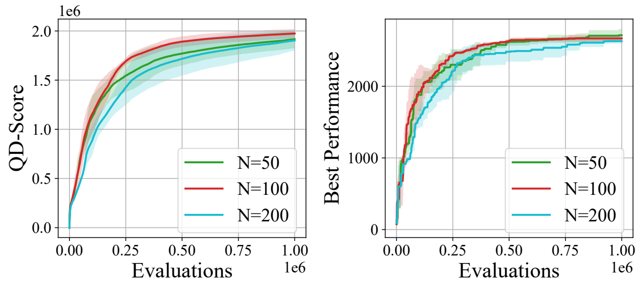

Figure 6: QD-Score and best performance for different values of number N of selected solutions in each generation on QD Hopper.

selection methods (i.e., NSLC, Curiosity and EDO-CS) have additional hyperparameters, e.g., Curiosity requires to decide the reward and penalty values of curiosity score. The ablation study on the number M of cells is provided in Appendix C.4.

5 Conclusion

The parent selection methods play an important role in QD algorithms. This paper analyzes why classical MO-based selection fails, and then proposes the NSS-based selection method by considering all directions in the behavior space. Experiments on synthetic functions and several complex tasks show the superiority of NSS. NSS can be incorporated into various QD algorithms, which can be beneficial in real-world applications. Improving the scalability of NSS w.r.t the number of behavior descriptor functions is an interesting future work.

References

[Allard *et al.*, 2022] Maxime Allard, Simón C Smith, Konstantinos Chatzilygeroudis, Bryan Lim, and Antoine Cully. Online damage recovery for physical robots with hierarchical quality-diversity. *arXiv:2210.09918*, 2022.

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

[Bhatt *et al.*, 2022] Varun Bhatt, Bryon Tjanaka, Matthew C. Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, New Orleans, LA, 2022.

[Chalumeau *et al.*, 2023] Felix Chalumeau, Raphael Boige, Bryan Lim, Valentin Macé, Maxime Allard, Arthur Flajolet, Antoine Cully, and Thomas Pierrot. Neuroevolution is a competitive alternative to reinforcement learning for skill discovery. In *The 11th International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 2023.

[Chatzilygeroudis *et al.*, 2021] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: A novel branch of stochastic optimization. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, pages 109–135. Springer, 2021.

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

[Cully *et al.*, 2015] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[Deb *et al.*, 2002] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[Deb, 2011] Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: An introduction. In *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pages 3–34. Springer, 2011.

[Do *et al.*, 2022] Anh Do, Mingyu Guo, Aneta Neumann, and Frank Neumann. Analysis of evolutionary diversity optimization for permutation problems. *ACM Transactions on Evolutionary Learning and Optimization*, 2(3):1–27, 2022.

[Ecoffet *et al.*, 2021] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[Eysenbach *et al.*, 2018] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *The 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[Flageat *et al.*, 2023] Manon Flageat, Felix Chalumeau, and Antoine Cully. Empirical analysis of PGA-MAP-Elites for neuroevolution in uncertain domains. *ACM Transactions on Evolutionary Learning and Optimization*, 2023.

[Fontaine and Nikolaidis, 2021] Matthew Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 10040–10052, Virtual, 2021.

[Fontaine *et al.*, 2021] Matthew C. Fontaine, Ruilin Liu, Ahmed Khalifa, Jignesh Modi, Julian Togelius, Amy K. Hoover, and Stefanos Nikolaidis. Illuminating mario scenes in the latent space of a generative adversarial network. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pages 5922–5930, Virtual, 2021.

[Kumar *et al.*, 2020] Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One solution is not all you need: Few-shot extrapolation via structured MaxEnt RL. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 8198–8210, Vancouver, Canada, 2020.

[Lehman and Stanley, 2011] Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th ACM Genetic and Evolutionary Computation Conference (GECCO)*, pages 211–218, Dublin, Ireland, 2011.

[Lupu *et al.*, 2021] Andrei Lupu, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 7204–7213, Virtual, 2021.

[Miao *et al.*, 2022] Jiayu Miao, Tianze Zhou, Kun Shao, Ming Zhou, Weinan Zhang, Jianye Hao, Yong Yu, and Jun Wang. Promoting quality and diversity in population-based reinforcement learning via hierarchical trajectory space exploration. In *Proceedings of the 39th IEEE International Conference on Robotics and Automation (ICRA)*, pages 7544–7550, Philadelphia, PA, 2022.

[Mouret and Clune, 2015] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv:1504.04909*, 2015.

[Nikfarjam *et al.*, 2022] Adel Nikfarjam, Amirhossein Moosavi, Aneta Neumann, and Frank Neumann. Computing high-quality solutions for the patient admission scheduling problem using evolutionary diversity optimisation. In *Proceedings of the 17th International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 250–264, Dortmund, Germany, 2022.

[Nilsson and Cully, 2021] Olle Nilsson and Antoine Cully. Policy gradient assisted MAP-Elites. In *Proceedings of the 23th ACM Genetic and Evolutionary Computation Conference (GECCO)*, page 866–875, Lille, France, 2021.

[Parker-Holder *et al.*, 2020] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M. Choromanski, and Stephen J.

Roberts. Effective diversity in population based reinforcement learning. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 18050–18062, Vancouver, Canada, 2020.

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

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

[Sheikh *et al.*, 2022] Hassam Sheikh, Kizza Frisbee, and Mariano Phielipp. DNS: Determinantal point process based neural network sampler for ensemble reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 19731–19746, Baltimore, MD, 2022.

[Shen *et al.*, 2020] Ruimin Shen, Yan Zheng, Jianye Hao, Zhaopeng Meng, Yingfeng Chen, Changjie Fan, and Yang Liu. Generating behavior-diverse game ais with evolutionary multi-objective deep reinforcement learning. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3371–3377, Yokohama, Japan, 2020.

[Tjanaka *et al.*, 2022] Bryon Tjanaka, Matthew C. Fontaine, Julian Togelius, and Stefanos Nikolaidis. Approximating gradients for differentiable quality diversity in reinforcement learning. In *Proceedings of the 24th ACM Genetic and Evolutionary Computation Conference (GECCO)*, page 1102–1111, Boston, MA, 2022.

[Villin *et al.*, 2021] Victor Villin, Naoki Masuyama, and Yusuke Nojima. Effects of different optimization formulations in evolutionary reinforcement learning on diverse behavior generation. In *Proceedings of the 7th IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–08, Orlando, FL, 2021.

[Wang *et al.*, 2022] Yutong Wang, Ke Xue, and Chao Qian. Evolutionary diversity optimization with clustering-based selection for reinforcement learning. In *The 10th International Conference on Learning Representations (ICLR)*, Virtual, 2022.

[Xue *et al.*, 2022] Ke Xue, Yutong Wang, Lei Yuan, Cong Guan, Chao Qian, and Yang Yu. Heterogeneous multi-agent zero-shot coordination by coevolution. *arXiv:2208.04957*, 2022.

[Yu *et al.*, 2023] Chao Yu, Jiaxuan Gao, Weilin Liu, Botian Xu, Hao Tang, Jiaqi Yang, Yu Wang, and Yi Wu. Learning zero-shot cooperation with humans, assuming humans are biased. In *The 11th International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 2023.

[Yuan *et al.*, 2023a] Lei Yuan, Feng Chen, Zongzhang Zhang, and Yang Yu. Communication-robust multi-agent learning by adaptable auxiliary multi-agent adversary generation. *arXiv:2305.05116*, 2023.

[Yuan *et al.*, 2023b] Lei Yuan, Ziqian Zhang, Ke Xue, Hao Yin, Feng Chen, Cong Guan, Lihe Li, Chao Qian, and Yang Yu. Robust multi-agent coordination via evolutionary generation of auxiliary adversarial attackers. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*, Washington, DC, 2023.

[Zhang *et al.*, 2023] Ziqian Zhang, Lei Yuan, Lihe Li, Ke Xue, Chengxing Jia, Cong Guan, Chao Qian, and Yang Yu. Fast teammate adaptation in the presence of sudden policy change. In *Preoceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI)*, Pittsburgh, PA, 2023.

[Zhou *et al.*, 2019] Zhi-Hua Zhou, Yang Yu, and Chao Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.