QUALITY-SIMILAR DIVERSITY VIA POPULATION BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diversity is a growing research topic in Reinforcement Learning (RL). Previous research on diversity has mainly focused on promoting diversity to encourage exploration and thereby improve quality (the cumulative reward), maximizing diversity subject to quality constraints, or jointly maximizing quality and diversity, known as the quality-diversity problem. In this work, we present the quality-similar diversity problem that features diversity among policies of *similar qualities*. In contrast to task-agnostic diversity, we focus on task-specific diversity defined by a set of user-specified Behavior Descriptors (BDs). A BD is a scalar function of a trajectory (e.g., the fire action rate for an Atari game), which delivers the type of diversity the user prefers. To derive the gradient of the user-specified diversity with respect to a policy, which is not trivially available, we introduce a set of BD estimators and connect it with the classical policy gradient theorem. Based on the diversity gradient, we develop a population-based RL algorithm to adaptively and efficiently optimize the population diversity at multiple quality levels throughout training. Extensive results on MuJoCo and Atari demonstrate that our algorithm significantly outperforms previous methods in terms of generating user-specified diverse policies across different quality levels (see Atari and MuJoCo videos).

1 INTRODUCTION

Existing research on policy diversity in deep Reinforcement Learning (RL) can be generally divided into three categories, according to the role diversity plays. The first category (Hong et al., 2018; Eysenbach et al., 2018; Conti et al., 2018; Parker-Holder et al., 2020; Kumar et al., 2020; Peng et al., 2020; Tang et al., 2020; Han & Sung, 2021; Chenghao et al., 2021; McKee et al., 2022) focuses on maximizing the final quality (the cumulative reward) of a policy, and policy diversity only serves as a means to better fulfill this goal via improving the efficiency of exploration. Therefore, the diversity measure is preferred to be task-agnostic as the knowledge of what type of task-specific diversity benefits the quality may not be accessible in most cases. The second category (Masood & Doshi-Velez, 2019; Zhang et al., 2019; Sun et al., 2020; Ghasemi et al., 2021; Zahavy et al., 2021; Zhou et al., 2022) is concerned with constrained optimization problems, where either diversity is optimized subject to quality constraints or vice-versa. Again, existing methods in this category have mainly focused on task-agnostic diversity, thereby the obtained diversity is often explained in hindsight, i.e., it is unknown what type of policy diversity to expect until the optimization is finished. The third category optimizes quality and diversity simultaneously, which is usually known as the Quality-Diversity (QD) method (Cully et al., 2015; Mouret & Clune, 2015; Pugh et al., 2016; Colas et al., 2020; Fontaine & Nikolaidis, 2021; Nilsson & Cully, 2021; Pierrot et al., 2022; Wang et al., 2021; Tjanaka et al., 2022). In contrast to task-agnostic diversity, most QD methods focus on task-specific diversity, where users are allowed to specify a set of interested Behavior Descriptors (BDs). A BD is a scalar function of a trajectory (i.e., the whole game episode) and thus does not have an analytical function form with respect to a single policy or state. Therefore, the gradient of a BD with respect to a policy is not trivially available, and this extends to the diversity measure defined on multiple BDs. As a result, previous QD methods (Cully et al., 2015; Mouret & Clune, 2015; Pugh et al., 2016) rely on black-box optimization techniques, such as evolutionary algorithms, to evolve a population of diverse policies. Some recent QD methods (Colas et al., 2020; Fontaine & Nikolaidis, 2021; Nilsson & Cully, 2021; Pierrot et al., 2022; Tjanaka et al., 2022) try to inject gradient information into the evolutionary optimization process.

In this work, we formulate the Quality-Similar Diversity (QSD) problem where the objective is to produce a set of diverse policies at multiple quality levels. We propose a new QD metric called the QSD score that clusters policies of similar qualities, and the diversity is evaluated at each quality level. In QSD problems, diverse policies of non-optimal qualities are also preferred, which directly meet practical needs in some real-world AI applications. For example, in the field of game AI (Shen et al., 2020; Ye et al., 2020; Fu et al., 2021), it is often desirable to provide diverse accompanying AIs whose qualities are matched to a beginner, an amateur, and a master, respectively. Besides, measuring the diversity between a beginner and a master would be of little interest. The QSD problem also connects with adaptive curricula (Wang et al., 2019; Team et al., 2021; Parker-Holder et al., 2022), where the environment gradually increases curriculum levels from simple to complex. Optimizing the intermediate diversity at non-optimal quality levels helps a faster and better convergence of the agent's capabilities than training directly at a complex curriculum level.

Moreover, the ability to generate task-specific diversity is superior and supplementary to task-agnostic diversity when the user has a clear preference for the type of diversity in practice. For example, diverse hand gestures are of no interest if the user only needs gait diversity in robot locomotion tasks. Hence, in this work, we optimize an explicit diversity measure function defined on several user-specified BDs, as opposed to the non-differentiable cell coverage percentage in most QD methods. To the best of our knowledge, none of existing methods has obtained the exact gradient of a user-specified BD (defined on trajectories) with respect to a policy, nor has any derived an unbiased estimation of this gradient using state-action samples. In particular, the diversity gradient is approximated by generating samples in the policy parameter space (Colas et al., 2020; Tjanaka et al., 2022), or simply assumed in Fontaine & Nikolaidis (2021), which might not hold in many real-world situations. A set of 'state' BDs (essentially a type of intrinsic reward) are introduced in Pierrot et al. (2022), expecting that a positive correlation between state and trajectory BDs might suffice.

To fill this gap, we propose a set of *BD estimators* that predict the corresponding BD value for the current policy. Equipped with these BD estimators, we build on the policy gradient theorem (Sutton et al., 1999; Silver et al., 2014) to derive the gradient of user-specified BDs with respect to a policy for discrete or continuous actions. Based on the population-based training (PBT) (Jaderberg et al., 2017), we develop an RL diversity algorithm, named QSD-PBT, that leverages diversity gradient and adaptively adjusts diversity loss to preserve similar qualities of the population. QSD-PBT efficiently optimizes the diversity of multiple quality levels in a single run and outperforms previous methods in terms of the QSD score in both MuJoCo and Atari environments. Meanwhile, QSD-PBT demonstrates strong abilities in achieving user-specified diversity by discovering visually distinct policies across a variety of environments. To summarize, the contributions of this work are as follows:

- We formulate the Quality-Similar Diversity (QSD) problem and propose a new performance metric.
- We derive the gradient of user-specified BDs defined on trajectories with respect to a policy.
- We develop a population-based RL algorithm that efficiently optimizes the diversity of multiple quality levels in a single run.

2 Problem Definition

We focus on the episodic Markov Decision Processes (MDPs), which can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ stand for the state space and action space respectively. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the environment transition function, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the expected reward function. A policy $\pi(s)$ maps a state s to a probability distribution over $\mathcal{A}$. A trajectory τ is a state-action sequence $[s_0, a_0, s_1, a_1, ..., s_T]$, which is obtained by executing a policy from the initial step $t = 0$ to the terminal step T in the environment. The objective of RL is to find a policy π that maximizes its expected cumulative rewards (also known as the quality in this work: $J(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)]$, where $R(\tau) = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$ is the return of a trajectory, and $\gamma \in [0, 1]$ is the discount factor. The state value function $V^\pi(s) = \mathbb{E}[\sum_{t=i}^{T} \gamma^{t-i} r(s_t, a_t)|s_i = s]$ measures the quality following π from state s. In particular, we define $V^\pi(s_T) = 0$. The Q-function $Q^\pi(s, a) = \mathbb{E}[\sum_{t=i}^{T} \gamma^{t-i} r(s_t, a_t)|s_i = s, a_i = a]$ measures the quality following π after taking action a in state s.

Behavior descriptor. We would like to maximize a task-specific diversity measure at different quality levels in this work. Hence, users are allowed to specify a set of BDs, $b_i(\tau)$ $(1 \leq i \leq L)$

that reflect the type of interested policy diversity. $b_i(\tau)$ is an evaluation function of a trajectory that is finite and easy to be implemented. For instance, a BD could be the ratio between left and right movements in the trajectory of Atari MsPacMan. The trajectory BD is a general form and can be simplified to state or action BD when the user is only interested in certain states or actions of the environment, e.g., the terminal position in MsPacMan. Accordingly, the BD value of a policy π_θ (parameterized using θ) is defined as $B_i(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[b_i(\tau)]$. We denote all the BD values of a policy π_θ by $\mathcal{B}(\pi_\theta) = [B_1(\pi_\theta), B_2(\pi_\theta), ..., B_L(\pi_\theta)]$.

Diversity measure function. Since BD values of a policy form an L-dimensional vector, we need a function f that measures the overall diversity of the population as a scalar: $\text{Div}(\Pi) := f(\mathcal{B}(\pi_{\theta_1}), \mathcal{B}(\pi_{\theta_2}), ..., \mathcal{B}(\pi_{\theta_N}))$, where $\Pi = \{\pi_{\theta_j} | 1 \leq j \leq N\}$ denotes a set of policies. The choice of the f should follow these properties: (1) f should be bounded and non-negative for easy implementation; (2) Since we do not define the order of an agent in the population, f should be invariant of any permutation of the policies; (3) f should be differentiable such that we can derive its gradient. Two measure functions are recommended and the formulations are detailed in Section C.2: the mean of all pair-wise euclidean distances, and the Determinantal Point Process (DPP) (Parker-Holder et al., 2020).

Quality-similar diversity score. Note that the quality of a policy for a task is usually real-valued and one-dimensional, obtaining a diverse set of policies at every possible quality level would require an infinite number of policies. Hence, we approximately evaluate an algorithm's QSD performance by partitioning the obtained quality range into M disjoint intervals, and only the diversity of policies within the same quality interval is evaluated by f. The QSD score is defined as follows:

$$\text{QSD score} := \sum_{m=1}^{M} \text{Div}(\Pi_m), \tag{1}$$

where Π_m denotes the set of policies obtained throughout training with qualities in the m-th interval. Note that we can put a weight for each interval in Equation 1, e.g., a user may focus on the diversity of policies that have medium-to-high qualities, then only these intervals in the quality range are considered in the QSD score.

3 Population-based RL for Quality-Similar Diversity

The gradient of the user-specified BD with respect to a policy is not tractable since it is defined on a trajectory. Hence, a possible solution to our QSD problem would be adapting derivative-free methods, such as conventional QD methods (Mouret & Clune, 2015). However, derivative-free methods often scale poorly with large-scale neural networks that are necessary to handle complex inputs, such as the image feature in Atari. Note that the form in our definition of user-specified BD is similar to the quality, it would be feasible to directly derive the diversity gradient using the policy gradient theorem (Sutton et al., 1999). Given the diversity gradient, there exist two choices as how to obtain a diverse population: sequential training (Zhang et al., 2019; Zahavy et al., 2021; Zhou et al., 2022) and population-based training (Jaderberg et al., 2017; Jung et al., 2019; Parker-Holder et al., 2020). The population-based training is more appropriate in our case, because the diversity measure is defined on a population that can not be factorized into incremental diversity settings as in Zhang et al. (2019); Zahavy et al. (2021); Zhou et al. (2022). Based on the above discussion, we develop an efficient population-based RL algorithm, named QSD-PBT, for optimizing user-specified diversity BDs across different quality levels. Each component of QSD-PBT is elaborated in the following.

3.1 Deriving the Diversity Gradient

Using the chain rule, we can write the gradient of the diversity of a population $\Pi = \{\pi_{\theta_j} | 1 \leq j \leq N\}$ with respect to the policy θ_j (without loss of generality, we use θ hereafter) as:

$$\frac{\partial \text{Div}(\Pi)}{\partial \theta} = \sum_{i=1}^{L} \frac{\partial f}{\partial B_i(\pi_\theta)} \frac{\partial B_i(\pi_\theta)}{\partial \theta}. \tag{2}$$

The partial derivative of f with respect to $B_i(\pi_\theta)$ is easily obtained, as long as f is an explicit diversity measure function, such as the mean pair-wise distance or the determinant in DPP. Note

that in $B_i(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[b_i(\tau)]$, $b_i(\tau)$ is a scalar function evaluating the i-th user-specified BD with respect to a trajectory τ. Following the policy gradient theorem (Sutton et al., 1999), we have:

$$\frac{\partial B_i(\pi_\theta)}{\partial\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} b_i(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)]. \tag{3}$$

The gradient in Equation 3 is preferred to be estimated by samples $[s_t, a_t, s_{t+1}, \hat{b}_i(\tau)]$ in practice. To this end, a set of state and state-action BD estimators parameterized by ϕ_i are introduced:

$$V_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t; \phi_i) := \mathbb{E}_{a_t,\tau_{t+1:T}\sim\pi_\theta}[b_i(\tau)], \tag{4}$$

$$Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t; \phi_i) := \mathbb{E}_{\tau_{t+1:T}\sim\pi_\theta}[b_i(\tau)], \tag{5}$$

where $\tau_{i:j}$ denotes a segment of a trajectory starting from time step i to j: $\tau_{i:j} = [s_i, a_i, ..., s_j, a_j]$. It is worth noting that $b_i(\tau)$ can not be factorized into a sum of quantities at each time step. For this reason, $V_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t)$ at state s_t depends on the whole historical state-action sequence $\tau_{0:t-1}$. Similar to the advantage in RL, we define the BD advantage $A_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t) = Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t) - V_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t)$. Since $V_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t)$ is action-independent, the gradient in Equation 3 can be estimated using a mini-batch of samples $\{(\tau_{0:t-1}^{(k)}, s_t^{(k)}, a_t^{(k)}, \hat{A}_{B_i}^{(k)})\}_{k=1}^{K}$ as:

$$\frac{\partial B_i(\pi_\theta)}{\partial\theta} \approx \frac{1}{K}\sum_{k=1}^{K} \hat{A}_{B_i}^{(k)} \nabla_\theta \log \pi_\theta(a_t^{(k)}|s_t^{(k)}), \tag{6}$$

where the sampled BD advantage $\hat{A}_{B_i}^{(k)}$ can be estimated using the conventional Generalized Advantage Estimator (GAE) (Schulman et al., 2016) or more advanced methods, e.g., Direct Advantage Estimation (Pan et al., 2021). The above derives the diversity gradient for the discrete-action policy. Based on the deterministic policy gradient theorem (Silver et al., 2014), we have the following proposition for deterministic policy $\pi_\theta(s)$ in continuous action space.

Proposition 1. *In deterministic policy and continuous action space case, the derivative of $B_i(\pi_\theta)$ with respect to policy parameters θ is:*

$$\frac{\partial B_i(\pi_\theta)}{\partial\theta} = \sum_{t=0}^{T}\int_{s_{0:t}} p(s_0 \to s_t)\nabla_\theta \pi_\theta(s_t)\nabla_{a_t} Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t)|_{a_t=\pi_\theta(s_t)} \mathrm{d}s_{0:t}, \tag{7}$$

where $\int_{s_{0:t}}$ and $\mathrm{d}s_{0:t}$ are short for $\int_{s_0}\int_{s_1}...\int_{s_t}$ and $\mathrm{d}s_0\mathrm{d}s_1...\mathrm{d}s_t$ respectively, and $p(s_i \to s_j) = p(s_i)\prod_{k=i}^{j-1} p(s_{k+1}|s_k, \pi_\theta(s_k))$. Using a mini-batch of samples$\{(\tau_{0:t-1}^{(k)}, s_t^{(k)}, a_t^{(k)})\}_{k=1}^{K}$, we can develop an unbiased estimator of the gradient above:

$$\frac{\partial B_i(\pi_\theta)}{\partial\theta} \approx \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \pi_\theta(s_t^{(k)})\nabla_{a_t^{(k)}} Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}^{(k)}, s_t^{(k)}, a_t^{(k)})|_{a_t^{(k)}=\pi_\theta(s_t^{(k)})}. \tag{8}$$

The proof of proposition 1 is in Appendix A.1. Based on the proposition, at each training step, we can sample a mini-batch $\{(\tau_{0:t-1}^{(k)}, s_t^{(k)}, a_t^{(k)})\}_{k=1}^{K}$ to estimate the gradient. It is worth noting that the trajectory $\tau_{0:t-1}$ may be a quite long sequence, e.g., more than 10k state-action pairs in Atari games. Hence, a feature extractor is needed to encode the trajectory. As for training BD estimators, We use the final BD value of a trajectory as the target and apply the mean squared error loss function. More implementation details are provided in Appendix C.4.

3.2 Overall Training Scheme

In the QSD problem, we focus on maximizing the diversity at different quality levels. Previous methods applied to QD problems such as the constrained optimization subject to some quality constraints are inconsistent with our objective. Another possible solution is to combine the quality loss and the diversity loss with a coefficient and adjust the coefficient using some online learning algorithms, such as bandits (Parker-Holder et al., 2020). However, it is not obvious what the online learning objective should be and how to adapt the coefficient optimally. To solve the QSD problem, we start from dealing with the sub-problem of QSD, i.e., maximizing the diversity at one quality level.

Optimizing diversity at one quality level. Considering a population with N policies, the diversity and the average quality of the population are $\mathrm{Div}(\Pi)$ and $\frac{1}{N}\sum_{j=1}^{N} J(\pi_{\theta_j})$ respectively. Each sub-problem corresponds to optimizing a combined quality loss and diversity loss with a fixed coefficient λ, saying the target weight λ_∞. Instead of training with λ_∞ from start to end, we let λ start from a large initial value λ_0 and gradually decay to λ_∞ at the end of training. Specifically, at each training step t, the combined loss for the population is $L_t(\Pi) = -\frac{1}{N}\sum_{j=1}^{N} J(\pi_{\theta_j}) - \lambda_t \mathrm{Div}(\Pi)$, where we require $\lambda_{t+1} < \lambda_t$ and $\lim_{t\to\infty} \lambda_t = \lambda_\infty$. This is motivated by a general observation that exploration is important in population-based RL training, and a larger λ focuses more on the diversity and thus helps exploration. Apart from encouraging exploration throughout training, our decaying method preserves the convergence property under some assumptions, which is proved in the following.

Proposition 2. *Let $f(x) : \mathbb{R}^n \to \mathbb{R}, g(x) : \mathbb{R}^n \to \mathbb{R}$ be Lipschitz smooth, convex, bounded functions with derivative bounded, and a positive decreasing series $\{\lambda_t\}_{t=0}^{\infty}$ converges to λ. Denote $h_t(x) = f(x) + \lambda_t g(x)$. Then using the gradient descent algorithm and choosing a proper step size, the algorithm will converge to the global minimum of $f(x) + \lambda g(x)$.*

The proof is in Appendix A.2. We can see that the decaying method converges in convex and Lipschitz smooth cases. In non-convex cases, we validate it experimentally in Section 4 and Appendix B.2.

Preserving the quality similarity. Constraining only the average quality of the population at certain levels does not guarantee similar qualities in the population. For instance, quality values $\{1, 5, 9\}$ have the same average quality as quality values $\{4, 5, 6\}$, but the agent quality in the latter population is more similar to each other. To encourage quality similarity among agents in a population, we distribute λ_t to each agent in the population, denoted as $\lambda_t^{(j)}$ for the j-th agent, and adapt each $\lambda_t^{(j)}$ during training. The loss function for the j-th agent (policy) is $L_t^{(j)}(\pi_{\theta_j}) = -J(\pi_{\theta_j}) - \lambda_t^{(j)}\mathrm{Div}(\Pi)$. Specifically, policies with better qualities should focus more on diversity optimization, and vice-versa. Following the discussion above, the scheme of $\lambda_t^{(j)}$ throughout the training process is designed as $\lambda_t^{(j)} = \lambda_\infty + (\lambda_0 - \lambda_\infty)\exp(-\frac{t}{t_0} \cdot \frac{\max_j R_t^{(j)}}{R_t^{(j)}})$, where λ_0 is an initial coefficient. t_0 indicates the preferred decay step, and t is the current training step. $R_t^{(j)}$ is the evaluation of the current quality of the j-th policy in the population. We assume that the quality of any policy is non-negative and upper bounded. As a result, we have $\lim_{t\to\infty} \lambda_t^{(j)} = \lambda_\infty$.

Optimizing diversity across multiple quality levels. Once a target λ_∞ is designated, the objective function $L_t(\Pi)$ corresponds to maximizing diversity at a certain quality level. Since our goal in QSD is to obtain policy diversity at multiple quality levels, a straightforward way is to optimize the objective function $L_t(\Pi)$ multiple times with a set of target weights $\{\lambda_{\infty,h}\}_{h=1}^{H}$. Alternatively, we could obtain policy diversity at multiple quality levels in a single run. Given two target weights $\lambda_{\infty,1} > \lambda_{\infty,2}$, when we have finished the optimization with $\lambda_{\infty,1}$, we can continue the optimization from $\lambda_{\infty,1}$ to solve the quality level targeted by $\lambda_{\infty,2}$, rather than re-initializing the model and training from the initial λ_0. This can be extended to multiple target weights, if λ is decayed slowly enough from a large value λ_0 to 0. In practice, we found this much more efficient than independent training with multiple target values $\{\lambda_{\infty,h}\}_{h=1}^{H}$. As a result, we apply a single training with λ decaying from λ_0 to 0 to integrally solve the QSD problem, which gives the overall training loss for the j-th policy at time step t as:

$$L_t^{(j)}(\pi_{\theta_j}) = -J(\pi_{\theta_j}) - \lambda_t^{(j)}\mathrm{Div}(\Pi), \tag{9}$$

with $\lambda_t^{(j)} = \lambda_0 \cdot \exp(-\frac{t}{t_0} \cdot \frac{\max_j R_t^{(j)}}{R_t^{(j)}})$. We employ the PPO (Schulman et al., 2017) backbone for discrete actions, which is widely adopted in the game AI area (Berner et al., 2019; Ye et al., 2020). As for scenarios with continuous action space, the TD3 (Fujimoto et al., 2018) algorithm is applied as the backbone. We term our diversity optimization along with population-based RL backbones as QSD-PBT. The pseudocode is given in Appendix C.5.

4 Experiments

The effectiveness of QSD-PBT is validated on both MuJoCo (Brockman et al., 2016) continuous control tasks and Atari games (Bellemare et al., 2013) with discrete action spaces. We also justify the need for policy diversity at different quality levels in multiagent settings in Appendix B.1. For

the diversity measure, we use the mean of pair-wise Euclidean distance between the BD vectors of two policies. For the specification of BDs, we follow common practices in MuJoCo tasks by incorporating the built-in joint torques as BDs, similar to Parker-Holder et al. (2020). For Atari games, we estimate the advantage by the widely-adopted GAE method and follow the PPO Schulman et al. (2017) implementation that normalizes the advantage to make the training more robust. We design general-proposed BDs from the perspective of human players for the five Atari games. It is worth noting that the design of all the BDs does not unfairly favor any particular algorithm. Hence, the comparison among different methods is fair and unbiased (also illustrated in Appendix B.4). We compare QSD-PBT with independent PPO (Schulman et al., 2017) (driven only by the quality), two QD-style methods: QD-PG (Pierrot et al., 2022) (diversity gradient by 'state' BDs) and EDO-CS (Wang et al., 2021) (diversity gradient by evolution strategies), and two population-based RL algorithms: PBT (Jaderberg et al., 2017) (driven only by the quality) and DvD (Parker-Holder et al., 2020) (driven by both the quality and task-agnostic diversity).

For each environment, the range of quality is firstly estimated by training a state-of-the-art quality-driven method (TD3 for MuJoCo tasks and PPO for Atari games) and then partitioned into $M = 10$ disjoint intervals of equal scope. The highest quality R_{max} achieved by TD3 or PPO is considered as the 'optimal' quality and shared by all the comparing methods. To calculate the QSD score for each method, we save a number of candidate policies within each quality interval during training. The time overhead for each method is estimated by the number of training steps till the average quality of the population reaches a near-optimal quality ($0.9R_{max}$). Error bars plotted in the figures or standard deviations presented in the tables are obtained using 5 independent runs. The population size N is 8 for MuJoCo experiments and 10 for Atari experiments. Other implementation details are presented in Appendix C. Additional results that demonstrate the versatility of QSD-PBT for different user-specified BDs and diversity measure functions are included in Appendix B.4 and B.2.2.

4.1 MuJoCo Tasks

Three tasks from the MuJoCo environment are selected: Humanoid, Hopper, and Walker2d. We define two types of BDs: (1) **Scoring speed** is the final score divided by the number of time steps used, indicating how fast the agent scores. (2) **Built-in joint torques** are the average torques applied to the hinge joints over a trajectory, which have a dimension of 17, 3, and 6 respectively.

Table 1: The QSD scores on MuJoCo tasks. #step denotes the time overhead of each method.

MuJoCo Tasks		QD-PG	EDO-CS	PBT	DvD-TD3	QSD-PBT (Ours)
Humanoid	score > 0%	1.78 ± 0.12	1.56 ± 0.13	1.20 ± 0.20	3.02 ± 0.10	$\mathbf{3.45 \pm 0.07}$
R_{max}: 5.5k	score > 60%	0.40 ± 0.02	0.62 ± 0.03	0.30 ± 0.15	1.13 ± 0.02	$\mathbf{1.39 \pm 0.03}$
	#step (k)	760 ± 160	782 ± 176	120 ± 1	107 ± 7	$\mathbf{90 \pm 15}$
Hopper	score > 0%	0.59 ± 0.01	0.72 ± 0.09	0.63 ± 0.03	0.80 ± 0.02	$\mathbf{1.26 \pm 0.08}$
R_{max}: 3k	score > 60%	0.08 ± 0.01	0.18 ± 0.02	0.17 ± 0.01	0.24 ± 0.01	$\mathbf{0.26 \pm 0.01}$
	#step (k)	222 ± 31	200 ± 27	$\mathbf{111 \pm 7}$	208 ± 27	150 ± 19
Walker2d	score > 0%	0.91 ± 0.04	0.86 ± 0.10	0.78 ± 0.03	1.68 ± 0.16	$\mathbf{2.28 \pm 0.05}$
R_{max}: 5k	score > 60%	0.07 ± 0.01	0.13 ± 0.06	0.19 ± 0.01	0.47 ± 0.04	$\mathbf{0.79 \pm 0.02}$
	#step (k)	350 ± 51	471 ± 102	$\mathbf{155 \pm 2}$	191 ± 2	186 ± 14

Table 1 shows each method's QSD score, where we also report the QSD score calculated using only intervals with high quality (above 60% of R_{max}). PBT gets the lowest 3 out of 6 QSD scores, indicating that methods considering only quality result in considerable degradation of diversity. QD-PG gets trapped in terms of quality, which results in a low QSD score. EDO-CS performs slightly better than QD-PG, which is consistent with the experimental results in Wang et al. (2021). QD-PG and EDO-CS are much more time-consuming than other RL-style methods. The performance of DvD-TD3 (Parker-Holder et al., 2020) comes as the second best, and QSD-PBT consistently achieves the highest QSD scores. Similar conclusions can be arrived at in Figure 1, where we plot the quality-diversity curves throughout training. One important reason for the superiority of QSD-PBT is the unbiased diversity gradient with respect to the user-specified diversity and the estimation using samples in the state-action space, while other methods are either biased (QD-PG using 'state' BDs and DvD-TD3 using task-agnostic BDs) or estimated using samples in the policy parameter space

(EDO-CS). Another reason is that the diversity is better exploited within each quality interval due to the adaptive diversity loss, which is further studied in Appendix B.2. Additional illustration of the diversity at different quality levels is in Figure 9.

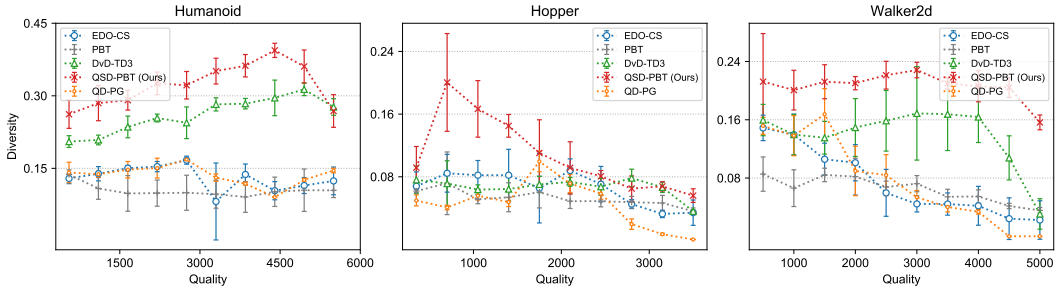


Figure 1: The quality similar diversity across 10 quality intervals on MuJoCo tasks.

4.2 Atari Games

We select five Atari games. MsPacMan and RiverRaid are PvE (i.e., single-agent) games where players obtain scores by achieving pre-designed goals. FishingDerby, IceHockey, and Boxing are PvP (i.e., multiagent) games where the final score is the score difference between the player and a built-in AI competitor. Five general-proposed BDs are constructed from the perspective of human players: (1) **Game time** is the number of time steps in an episode. (2) **Fire rate** is the frequency of *fire* action in an episode. (3) **Action continuity** measures the continuity of an agent's actions by counting the number of action changes, which is actually an AI version of APM (actions per minute, widely applied to human players). For example, this BD for the action sequence {*noop-noop-left-left-noop*} is 2. (4) Two orientation BDs (**left_right** and **up_down**) measure the preference of moving directions.

Table 2: The QSD scores on Atari games. #step denotes the time overhead of each method.

Atari Games			PPO	PBT	DvD-PPO	QSD-PBT (Ours)
PvE	MsPacMan R_{max}: 15k	score > 0%	1.28 ± 0.13	1.14 ± 0.09	1.40 ± 0.05	$\mathbf{2.47 \pm 0.06}$
		score > 60%	0.53 ± 0.06	0.42 ± 0.07	0.56 ± 0.04	$\mathbf{1.02 \pm 0.04}$
		#step (k)	456 ± 86	$\mathbf{269 \pm 52}$	415 ± 52	902 ± 108
	RiverRaid R_{max}: 15k	score > 0%	0.92 ± 0.12	0.84 ± 0.08	1.32 ± 0.12	$\mathbf{2.66 \pm 0.13}$
		score > 60%	0.43 ± 0.08	0.38 ± 0.05	0.60 ± 0.05	$\mathbf{1.32 \pm 0.04}$
		#step (k)	204 ± 31	$\mathbf{154 \pm 36}$	352 ± 144	734 ± 183
PvP	FishingDerby R_{max}: 48	score > 0%	1.60 ± 0.17	1.55 ± 0.13	2.32 ± 0.08	$\mathbf{3.64 \pm 0.12}$
		score > 60%	0.78 ± 0.09	0.76 ± 0.06	1.01 ± 0.06	$\mathbf{1.68 \pm 0.05}$
		#step (k)	262 ± 138	$\mathbf{109 \pm 25}$	168 ± 87	242 ± 25
	IceHockey R_{max}: 40	score > 0%	1.68 ± 0.16	1.04 ± 0.14	2.74 ± 0.30	$\mathbf{4.81 \pm 0.27}$
		score > 60%	0.81 ± 0.05	0.43 ± 0.07	1.13 ± 0.12	$\mathbf{1.92 \pm 0.08}$
		#step (k)	551 ± 100	520 ± 152	$\mathbf{443 \pm 42}$	404 ± 61
	Boxing R_{max}: 100	score > 0%	0.56 ± 0.08	0.69 ± 0.07	0.92 ± 0.28	$\mathbf{1.28 \pm 0.17}$
		score > 60%	0.24 ± 0.07	0.30 ± 0.05	0.40 ± 0.13	$\mathbf{0.59 \pm 0.09}$
		#step (k)	24 ± 4	$\mathbf{23 \pm 1}$	29 ± 7	27 ± 3

In Atari games we find that the quality of QD-PG and EDO-CS algorithms reach much lower levels than other PPO-based methods, therefore we have not included them for comparison and introduced a PPO baseline. The results are summarized in Table 2 and Figure 2, most of which are consistent with that in the MuJoCo experiment. The exploitation strategy in PBT accelerates the training of quality at a considerable cost of diversity. DvD-PPO performs relatively well, even though it optimizes the task-agnostic diversity defined on action probabilities. Yet, the BDs for Atari games are more macroscopical than those for MuJoCo tasks, so the improvement of DvD-PPO over PBT is smaller in Atari games than that in MuJoCo tasks. Due to our diversity gradient and the adaptive diversity loss, QSD-PBT algorithm consistently achieves the highest QSD scores, though with more time overhead in MsPacMan and RiverRaid due to a relatively large diversity loss coefficient at the beginning.

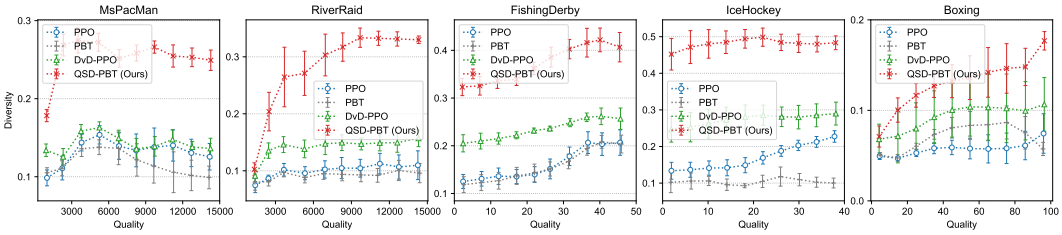


Figure 2: The quality similar diversity across 10 quality intervals on Atari games.

We further demonstrate the ability of QSD-PBT in generating user-specified diversity, using the two orientation diversity BDs in MsPacMan, as shown in Figure 3 and in the video. In this game, there are shortcuts connecting the left and right ends of the maze, which allows an agent to return to the middle of the maze via consistent (e.g., always left) moving directions. The orientation preference is a macroscopic BD that is significantly different from any task-agnostic diversity investigated in previous literature. As shown in Figure 3, QSD-PBT achieves diverse (in a user-specified way) and visually distinct policies with good quality while other comparing methods perform unsatisfactorily (including DvD-PPO which optimizes a task-agnostic diversity).

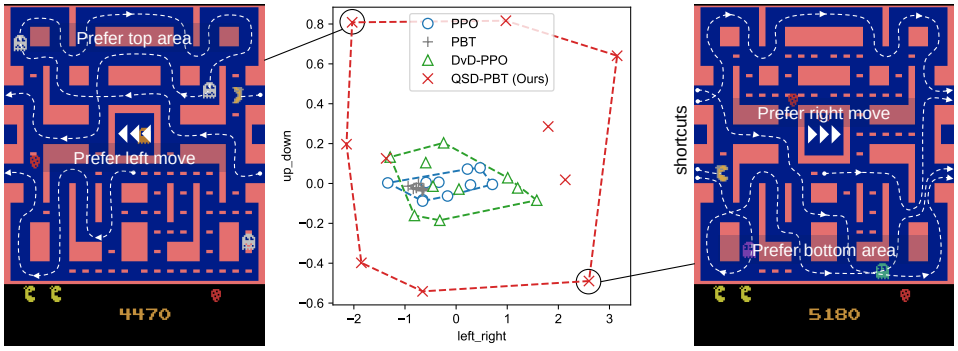


Figure 3: Visualization of orientation diversity in MsPacMan.

5 Related Work

Diversity as a means to improve quality. A diversity-driven approach for exploration (Hong et al., 2018) was proposed by adding a distance regularization between the current policy and a previous policy. Unsupervised learning of diverse policies was studied in Eysenbach et al. (2018) to serve as an effective pretraining mechanism for downstream RL tasks. Novelty search was hybridized with the OpenAI ES to improve exploration in sparse or deceptive deep RL tasks (Conti et al., 2018). A population determinant diversity measure (Parker-Holder et al., 2020) was proposed to improve exploration. Diverse behaviors were learned in order to effectively generalize to varying environments that are different from training (Kumar et al., 2020). A diversity-regularized collaborative exploration strategy was proposed in Peng et al. (2020). Reward randomization (Tang et al., 2020) was employed to discover diverse policies in multi-agent games with the aim of improving the final policy quality. Trajectory diversity was maximized for better zero-shot coordination in a collaborative multi-agent environment (Lupu et al., 2021). Diversity was studied in multi-agent open-ended learning (Liu et al., 2021; Perez-Nieves et al., 2021) to improve the final exploitability (another definition of quality).

Maximizing diversity subject to high quality constraints or vice-versa. A maximum mean discrepancy regularizer was proposed to produce a set of near-optimal policies having different distributions of trajectories (Masood & Doshi-Velez, 2019). A two-objective update technique (Zhang et al., 2019) was developed for sequentially obtaining a set of novel policies, each of which solves a given task in the meantime executing distinct action sequences. A method based on the Frank-Wolfe algorithm (Frank & Wolfe, 1956) was introduced to compute a set of diverse and near-optimal policies

(Ghasemi et al., 2021). A set of diverse policies in the space of successor features (Barreto et al., 2017) were sequentially obtained by solving a line of constrained MDPs (Zahavy et al., 2021), where an intrinsic diversity reward was maximized subject to a quality constraint. A reward-switching technique was recently proposed (Zhou et al., 2022) to discover a diverse set of high-quality policies by sequentially solving a novelty-constrained optimization problem for the current policy.

Maximizing both quality and diversity via QD-style methods. QD algorithms (Pugh et al., 2016; Cully & Demiris, 2017) are a type of evolutionary algorithms, where the goal is to evolve a set of diverse and high-quality solutions in a single run. A representative QD algorithm is MAP-Elites (Cully et al., 2015; Mouret & Clune, 2015). In order to scale to large policy neural networks, several recent attempts have been made to combine QD with policy gradient (Cideron et al., 2020; Nilsson & Cully, 2021; Pierrot et al., 2022) or ES (Colas et al., 2020; Wang et al., 2021). Evolutionary multi-objective deep reinforcement learning was employed (Shen et al., 2020) to generate behavior-diverse game AIs. A kernel-based method with Stein variational gradient descent was proposed (Gangwani et al., 2020) for training a set of QD policies. Assuming that both the quality and the BD are fully differentiable (which is generally not true in the RL settings), a new MAP-Elites algorithm was developed in Fontaine & Nikolaidis (2021). A subsequent work (Tjanaka et al., 2022) extended differentiable QD (Fontaine & Nikolaidis, 2021) to RL settings, where the quality and BD gradients were estimated using TD3 and ES respectively.

6 Discussion and Conclusion

In this work, motivated by the practical need in generating task-specific diverse policies of similar qualities, we formulate the QSD problem and a new performance metric called the QSD score. For the first time, we derive the gradient of the user-defined diversity measure with respect to a policy and approximate the gradient using samples in the state action space (as opposed to the policy parameter space). Based on our diversity gradient, an efficient population-based RL algorithm (i.e., QSD-PBT) is then developed and has demonstrated strong performance, in terms of maximizing user-specified diversity across different quality levels, on both MuJoCo and Atari.

BD definition. We are aware in some real-world situations it might be non-trivial to express the user-intended diversity via a set of BDs but relatively easy via a similarity function $\mathrm{Sim}(\tau_{\mathrm{i}}, \tau_{\mathrm{j}})$ between two trajectories. $\mathrm{Sim}(\tau_{\mathrm{i}}, \tau_{\mathrm{j}})$ is more general than BDs, since we can derive $\mathrm{Sim}(\tau_{\mathrm{i}}, \tau_{\mathrm{j}})$ using BDs but the opposite is not true. We look forward to extending QSD-PBT to such and more general situations.

BD estimator. In order to estimate our diversity gradient, we need to train the state BD (Equation 4) or state-action BD estimators (Equation 5). Note that both BD estimators depend on previous state-action sequence $\tau_{0:t-1}$, which might cause problems in very long trajectories. It is currently handled by applying LSTM models (in MuJoCo) or sufficient statistics (in Atari). More advanced techniques, such as attention (Vaswani et al., 2017), may be needed for more accurate estimation.

Population-based training. We use a separate neural network for each agent in the population, which increases the overall training time and memory overhead linearly with the population size N. A more efficient approach may be sharing most of the feature extraction part, e.g., the 3-layer convolution model in DQN, for all agents and building separate policy and value heads for each agent in the population. This implementation is connected with the ensemble and multi-task learning (An et al., 2021; Flet-Berliac & Preux, 2019). From experience in previous research, parameter sharing accelerates training convergence yet in our case degrades the diversity of PPO and PBT, since they are optimized without explicit diversity objectives. For fair comparisons, parameter sharing is not implemented in this paper but is recommended in practice.

Quality and diversity optimization. In this paper, we break the QSD problem into a set of sub-problems (each corresponding to maximizing diversity at a certain quality level) and solve them integrally in a single run, and the resulting algorithm is termed QSD-PBT. In spite of achieving better empirical performance than other baselines, whether QSD-PBT is the most efficient method in optimizing the QSD score defined in Equation 1 is unclear. One possible solution is combining the strength of our diversity gradient (Equation 6 and 8) with existing QD-style methods. Another is regarding the QSD problem as a Pareto front optimization problem and applying continuous exploration (Ma et al., 2020).

REFERENCES

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Li Chenghao, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Geoffrey Cideron, Thomas Pierrot, Nicolas Perrin, Karim Beguir, and Olivier Sigaud. Qd-rl: Efficient mixing of quality and diversity in reinforcement learning. *arXiv preprint arXiv:2006.08505*, 2020.

Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. Scaling map-elites to deep neuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 67–75, 2020.

Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in neural information processing systems*, 31, 2018.

Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

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

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

Jiajun Fan, Changnan Xiao, and Yue Huang. Gdi: Rethinking what makes reinforcement learning different from supervised learning. *arXiv preprint arXiv:2106.06232*, 2021.

Yannis Flet-Berliac and Philippe Preux. Merl: Multi-head reinforcement learning. *arXiv preprint arXiv:1909.11939*, 2019.

Matthew Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. *Advances in Neural Information Processing Systems*, 34, 2021.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Haobo Fu, Weiming Liu, Shuang Wu, Yijia Wang, Tao Yang, Kai Li, Junliang Xing, Bin Li, Bo Ma, Qiang Fu, et al. Actor-critic policy optimization in a large-scale imperfect-information game. In *International Conference on Learning Representations*, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Tanmay Gangwani, Jian Peng, and Yuan Zhou. Harnessing distribution ratio estimators for learning agents with quality and diversity. In *2020 Conference on Robot Learning (CoRL 2020)*, 2020.

Mahsa Ghasemi, Evan Scope Crafts, Bo Zhao, and Ufuk Topcu. Multiple plans are better than one: Diverse stochastic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pp. 140–148, 2021.

Seungyul Han and Youngchul Sung. Diversity actor-critic: Sample-aware entropy regularization for sample-efficient exploration. In *International Conference on Machine Learning*, pp. 4018–4029. PMLR, 2021.

Zhang-Wei Hong, Tzu-Yun Shann, Shih-Yang Su, Yi-Hsiang Chang, Tsu-Jui Fu, and Chun-Yi Lee. Diversity-driven exploration strategy for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Whiyoung Jung, Giseung Park, and Youngchul Sung. Population-guided parallel policy search for reinforcement learning. In *International Conference on Learning Representations*, 2019.

Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

Xiangyu Liu, Hangtian Jia, Ying Wen, Yujing Hu, Yingfeng Chen, Changjie Fan, Zhipeng Hu, and Yaodong Yang. Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. *Advances in Neural Information Processing Systems*, 34:941–952, 2021.

Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*, pp. 7204–7213. PMLR, 2021.

Pingchuan Ma, Tao Du, and Wojciech Matusik. Efficient continuous pareto exploration in multi-task learning. In *International Conference on Machine Learning*, pp. 6522–6531. PMLR, 2020.

Muhammad A Masood and Finale Doshi-Velez. Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies. *arXiv preprint arXiv:1906.00088*, 2019.

Kevin R McKee, Joel Z Leibo, Charlie Beattie, and Richard Everett. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–16, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

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

Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 866–875, 2021.

Hsiao-Ru Pan, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf. Direct advantage estimation. *arXiv preprint arXiv:2109.06093*, 2021.

Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. *arXiv preprint arXiv:2203.01302*, 2022.

Zhenghao Peng, Hao Sun, and Bolei Zhou. Non-local policy optimization via diversity-regularized collaborative exploration. *arXiv preprint arXiv:2006.07781*, 2020.

Nicolas Perez-Nieves, Yaodong Yang, Oliver Slumbers, David H Mguni, Ying Wen, and Jun Wang. Modelling behavioural diversity for learning in open-ended games. In *International Conference on Machine Learning*, pp. 8514–8524. PMLR, 2021.

Thomas Pierrot, Valentin Macé, Felix Chalumeau, Arthur Flajolet, Geoffrey Cideron, Karim Beguir, Antoine Cully, Olivier Sigaud, and Nicolas Perrin-Gilbert. Diversity policy gradient for sample efficient quality-diversity optimization. In *ICLR Workshop on Agent Learning in Open-Endedness*, 2022.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations, (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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 *IJCAI*, pp. 3371–3377, 2020.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. PMLR, 2014.

Hao Sun, Zhenghao Peng, Bo Dai, Jian Guo, Dahua Lin, and Bolei Zhou. Novel policy seeking with constrained optimization. *arXiv preprint arXiv:2005.10696*, 2020.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Zhenggang Tang, Chao Yu, Boyuan Chen, Huazhe Xu, Xiaolong Wang, Fei Fang, Simon Shaolei Du, Yu Wang, and Yi Wu. Discovering diverse multi-agent strategic behavior via reward randomization. In *International Conference on Learning Representations*, 2020.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Bryon Tjanaka, Matthew C Fontaine, Julian Togelius, and Stefanos Nikolaidis. Approximating gradients for differentiable quality diversity in reinforcement learning. *arXiv preprint arXiv:2202.03666*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Poet: open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 142–151, 2019.

Yutong Wang, Ke Xue, and Chao Qian. Evolutionary diversity optimization with clustering-based selection for reinforcement learning. In *International Conference on Learning Representations*, 2021.

Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:621–632, 2020.

Tom Zahavy, Brendan O'Donoghue, Andre Barreto, Sebastian Flennerhag, Volodymyr Mnih, and Satinder Singh. Discovering diverse nearly optimal policies with successor features. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

Yunbo Zhang, Wenhao Yu, and Greg Turk. Learning novel policies for tasks. In *International Conference on Machine Learning*, pp. 7483–7492. PMLR, 2019.

Zihan Zhou, Wei Fu, Bingliang Zhang, and Yi Wu. Continuously discovering novel strategies via reward-switching policy optimization. *arXiv preprint arXiv:2204.02246*, 2022.

A THEORY ANALYSIS

A.1 PROOF OF PROPOSITION 1

Proof. We prove the correctness of Equation 7 and that Estimation 8 is an unbiased estimation of the gradient in Equation 7 with a constant scale. We denote a segmentation of a trajectory starting from time step i to j as $\tau_{i:j} := [s_i, a_i, ..., s_j, a_j]$, with specially $\tau_{-1} := \varnothing$ and $\tau_0 := [s_0, a_0]$. Recall the definitions of state BD estimator and state-action BD estimator:

$$V_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t) := \mathbb{E}_{a_t, \tau_{t+1:T} \sim \pi_\theta}[b_i(\tau)], \tag{10}$$

$$Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t) := \mathbb{E}_{\tau_{t+1:T} \sim \pi_\theta}[b_i(\tau)]. \tag{11}$$

We consider a determinate policy for continuous actions, and the output of the policy $\pi_\theta(s_t)$ is a scalar, i.e., $a_t = \pi_\theta(s_t)$. As a result, the following two equations hold between the state BD estimator and the state-action BD estimator:

$$Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t) = \int_{s_{t+1}} p(s_{t+1}|s_t, \pi_\theta(s_t)) V_{B_i}^{\pi_\theta}(\tau_{0:t}, s_{t+1}) \mathrm{d}s_{t+1}, \tag{12}$$

$$V_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t) = Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, \pi_\theta(s_t)), \tag{13}$$

where $p(s_{t+1}|s_t, \pi_\theta(s_t))$ is the transition probability from s_t to s_{t+1} after taking the action $\pi_\theta(s_t)$. We use $p^\pi(s_i \to s_j)$ to represent the probability $p(s_i) \prod_{k=i}^{j-1} p(s_{k+1}|s_k, \pi_\theta(s_k))$. We omit the index π for clarity in the following. Note that the BD of a policy can be written as:

$$B_i(\pi_\theta) = \int_{s_0} p(s_0) Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, \pi_\theta(s_0)) \mathrm{d}s_0, \tag{14}$$

and the gradient is:

$$\frac{\partial B_i(\pi_\theta)}{\partial \theta} = \int_{s_0} p(s_0) \nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, \pi_\theta(s_0)) \mathrm{d}s_0. \tag{15}$$

Further, we have:

$$\begin{aligned}
&\nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, \pi_\theta(s_0)) = \nabla_\theta \int_{s_1} p(s_1|s_0, \pi_\theta(s_0)) V_{B_i}^{\pi_\theta}(\tau_0, s_1) \mathrm{d}s_1 \\
=&\int_{s_1} \nabla_\theta \pi_\theta(s_0) \nabla_{a_0} p(s_1|s_0, a_0)|_{a_0=\pi_\theta(s_0)} V_{B_i}^{\pi_\theta}(\tau_0, s_1) \mathrm{d}s_1 + \int_{s_1} p(s_1|s_0, \pi_\theta(s_0)) \nabla_\theta V_{B_i}^{\pi_\theta}(\tau_0, s_1) \mathrm{d}s_1 \\
=&\nabla_\theta \pi_\theta(s_0) \int_{s_1} \nabla_{a_0} p(s_1|s_0, a_0)|_{a_0=\pi_\theta(s_0)} V_{B_i}^{\pi_\theta}(\tau_0, s_1) \mathrm{d}s_1 + \int_{s_1} p(s_1|s_0, \pi_\theta(s_0)) \nabla_\theta V_{B_i}^{\pi_\theta}(\tau_0, s_1) \mathrm{d}s_1 \\
=&\nabla_\theta \pi_\theta(s_0) \nabla_{a_0} Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, a_0)|_{a_0=\pi_\theta(s_0)} + \int_{s_1} p(s_1|s_0, \pi_\theta(s_0)) \nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_0, s_1, \pi_\theta(s_1)) \mathrm{d}s_1.
\end{aligned} \tag{16}$$

Equation 16 shows the relation between the gradient of the current state-action BD estimator and that of the next state-action BD estimator. Generalizing this result, we have

$$\begin{aligned}
\nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, \pi_\theta(s_t)) =& \nabla_\theta \pi_\theta(s_t) \nabla_{a_t} Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t)|_{a_t=\pi_\theta(s_t)} \\
&+ \int_{s_{t+1}} p(s_{t+1}|s_t, \pi_\theta(s_t)) \nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_{0:t}, s_{t+1}, \pi_\theta(s_{t+1})) \mathrm{d}s_{t+1}.
\end{aligned} \tag{17}$$

We can apply Equation 17 to Equation 15 recursively:

$$\begin{aligned}
\frac{\partial B_i(\pi_\theta)}{\partial \theta} &= \int_{s_0} p(s_0)\nabla_\theta \pi_\theta(s_0)\nabla_{a_0} Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, a_0)|_{a_0=\pi_\theta(s_0)}\mathrm{d}s_0 \\
&\quad + \int_{s_1}\int_{s_0} p(s_0)p(s_1|s_0,\pi_\theta(s_0))\nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_0, s_1, \pi_\theta(s_1))\mathrm{d}s_0\mathrm{d}s_1 \\
&= \int_{s_0} p(s_0)\nabla_\theta \pi_\theta(s_0)\nabla_{a_0} Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, a_0)|_{a_0=\pi_\theta(s_0)}\mathrm{d}s_0 \\
&\quad + \int_{s_{0:1}} p(s_0 \to s_1)\nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_0, s_1, \pi_\theta(s_1))\mathrm{d}s_{0:1} \\
&= \int_{s_0} p(s_0)\nabla_\theta \pi_\theta(s_0)\nabla_{a_0} Q_{B_i}^{\pi_\theta}(\tau_{-1}, s_0, a_0)|_{a_0=\pi_\theta(s_0)}\mathrm{d}s_0 \\
&\quad + \int_{s_{0:1}} p(s_0 \to s_1)\nabla_\theta \pi_\theta(s_1)\nabla_{a_1} Q_{B_i}^{\pi_\theta}(\tau_0, s_1, a_1)|_{a_1=\pi_\theta(s_1)}\mathrm{d}s_{0:1} \\
&\quad + \int_{s_{0:2}} p(s_0 \to s_2)\nabla_\theta Q_{B_i}^{\pi_\theta}(\tau_{0:1}, s_2, \pi_\theta(s_2))\mathrm{d}s_{0:2} \\
&= ... \\
&= \sum_{t=0}^{T}\int_{s_{0:t}} p(s_0 \to s_t)\nabla_\theta \pi_\theta(s_t)\nabla_{a_t} Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t)|_{a_t=\pi_\theta(s_t)}\mathrm{d}s_{0:t}, \qquad (18)
\end{aligned}$$

where $\int_{s_{0:t}}$ is short for $\int_{s_0}\int_{s_1}...\int_{s_t}$ and $\mathrm{d}s_{0:t}$ is short for $\mathrm{d}s_0\mathrm{d}s_1...\mathrm{d}s_t$. The first equality is according to Equation 16, and the successive equality is according to Equation 17. □

Proof. We further prove that with a constant scale, Equation 8 is an unbiased estimation of Equation 7. Note that in practice we obtain sampled trajectories by executing policy $\pi(\theta)$, and the samples in the trajectories are added into a replay buffer. As a result, a sample $(s_t, a_t, \tau_{0:t-1})$ is added into the replay buffer with a probability $p(s_0)\prod_{k=0}^{t-1} p(s_{k+1}|s_k, \pi_\theta(s_k))$, i.e., $p(s_0 \to s_t)$. In other words, the probability that the sample $(s_t, a_t, \tau_{0:t-1})$ is in the current mini-batch is proportional to $p(s_0 \to s_t)$, which we denote by $\frac{p(s_0\to s_t)}{C}$ (C is a constant). The expectation of the sampled gradient is:

$$\begin{aligned}
&\mathbb{E}\left[\nabla_\theta \pi_\theta(s_t)\nabla_{a_t} Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t)|_{a_t=\pi_\theta(s_t)}\right] \\
=&\sum_{t=0}^{T}\int_{s_{0:t}} \nabla_\theta \pi_\theta(s_t)\nabla_{a_t} Q_{B_i}^{\pi_\theta}(\tau_{0:t-1}, s_t, a_t)|_{a_t=\pi_\theta(s_t)}\frac{p(s_0 \to s_t)}{C}\mathrm{d}s_{0:t} \\
=&\frac{1}{C}\frac{\partial B_i(\pi_\theta)}{\partial \theta}, \qquad (19)
\end{aligned}$$

where the last equality is according to Equation 18. □

A.2 Proof of proposition 2

Proof. Denote $h(x) = f(x) + \lambda g(x)$. By assumption, we have $h(x)$ is Lipschitz smooth, i.e., there exists $L > 0$, s.t.

$$||\nabla h(x) - \nabla h(y)|| \leq L||x - y||, \forall x, y \in \mathbb{R}^n \qquad (20)$$

So,

$$\begin{aligned}
|h(y) - h(x) - \nabla h(x)^T(y-x)| &= |\int_0^1 \nabla h(x + t(y-x))^T(y-x)\mathrm{d}t - \nabla h(x)^T(y-x)| \\
&\leq \int_0^1 ||\nabla h(x + t(y-x))^T - \nabla h(x)^T|| \cdot ||y-x||\mathrm{d}t \\
&\leq \int_0^1 tL||y-x||^2\mathrm{d}t \\
&= \frac{L}{2}||y-x||^2 \qquad (21)
\end{aligned}$$

i.e.,

$$h(y) - h(x) \leq \nabla h(x)^T (y - x) + \frac{L}{2}||x - y||^2. \tag{22}$$

Since $g(x)$ is bounded, we can assume $g(x) > 0$. Otherwise, let $C = \min_x g(x)$, we rewrite $h_t(x) = f(x) + \lambda_t(g(x) - C) + \lambda_t C$ and perform the following analysis on $h_t(x)_{new} = f(x) + \lambda_t(g(x) - C)$ (note that both $h_t(x)_{new}$ and $h_t(x)$ have the same derivatives and minimums). Hence, for each fixed x, the series $\{h_t(x)\}_{t=0}^{\infty}$ is decreasing to $h(x)$.

Consider an optimizing algorithm which starts at x_0, and, at each time step t, updates x_t using the gradient descent $x_{t+1} = x_t - \eta \nabla h_t(x_t)$. By Equation 22, we have,

$$\begin{aligned} h_t(x_{t+1}) - h_t(x_t) &\leq \nabla h_t(x_t)^T (x_{t+1} - x_t) + \frac{L}{2}||x_{t+1} - x_t||^2 \\ &\leq -\eta ||\nabla h_t(x_t)||^2 + \frac{L\eta^2}{2}||\nabla h_t(x_t)||^2 \end{aligned} \tag{23}$$

Choosing $\eta = \frac{1}{L}$, we have, $h_t(x_{t+1}) \leq h_t(x_t)$, and since $h_{t+1}(x_{t+1}) \leq h_t(x_{t+1})$, we obtain $h_{t+1}(x_{t+1}) \leq h_t(x_t)$. Hence $\lim_{t\to\infty} h_t(x_t)$ exists ($h(x)$ is bounded). It is easy to prove that $\{h_t(x)\}_{t=0}^{\infty}$ converge to $h(x)$ uniformly, hence, $\lim_{t\to\infty} h_t(x_t) = \lim_{t\to\infty} h(x_t)$. We denote $M = \lim_{t\to\infty} h(x_t)$.

We claim that $\lim_{t\to\infty} \nabla h(x_t) = 0$, hence $M = \min_x h(x)$. We prove the claim by contradiction. Otherwise, there exists ϵ_0 satisfies that for any T, $\exists t_0 > T, s.t. ||\nabla h(x_{t_0})|| > \epsilon_0$.

Similar to Equation 23,

$$\begin{aligned} &h(x_{t+1}) - h(x_t) \leq \nabla h(x_t)^T (x_{t+1} - x_t) + \frac{L}{2}||x_{t+1} - x_t||^2 \\ &\leq -\eta \nabla h(x_t)^T \nabla h_t(x_t) + \frac{L\eta^2}{2}||\nabla h_t(x_t)||^2 \\ &= \eta(\nabla h_t(x_t)^T \nabla h_t(x_t) - \nabla h(x_t)^T \nabla h_t(x_t)) + (\frac{L\eta^2}{2} - \eta)||\nabla h_t(x_t)||^2 \\ &= \eta(\nabla h_t(x_t) - \nabla h(x_t))^T \nabla h_t(x_t) + -\frac{\eta}{2}||\nabla h_t(x_t)||^2. \end{aligned} \tag{24}$$

In the last equation, we choose $\eta = \frac{1}{L}$. Since $\lim_{t\to\infty} \nabla h_t(x) = \nabla h(x)$, we have $\lim_{t\to\infty} |(\nabla h_t(x_t) - \nabla h(x_t))^T \nabla h_t(x_t)| \leq \lim_{t\to\infty} ||\nabla h_t(x_t) - \nabla h(x_t)|| \cdot ||\nabla h_t(x_t)|| = 0$. Since whatever how large the T is, we can find $t_0 > T$ s.t. $||\nabla h(x_{t_0})|| > \epsilon_0$, so we can find $|h(x_{t_0+1}) - h(x_{t_0})| > \frac{\eta}{4}\epsilon_0$, which is contradict to the convergence of $\{h(x_t)\}_{t=0}^{\infty}$ (Cauchy principle of convergence).

Hence we prove the claim.

Finally, since $h(x)$ is convex, the algorithm will converge to a global minimum of $h(x)$. □

B Additional Experimental Results

B.1 Experimental Results on RoboSumo

The need of diversity among policies that have *similar qualities* is further justified on the multiagent MuJoCo benchmark RoboSumo (Al-Shedivat et al., 2017), where two agents wrestle against each other, as shown in Figure 4-left. We apply a fixed opponent obtained beforehand by TD3 and train agents with PBT, DVD-TD3 or QSD-PBT. The QSD scores are shown in Table 3. More importantly, using QSD-PBT, we obtain 16 agents for each of the three quality levels (Low, Medium, and High). We randomly sample agents within and across the three quality levels and make them compete with each other, for which the results (averaged over 100 episodes) of two game statistics (the game length and the score difference in an episode) are shown in Figure 4. Clearly, we can see that agents with similar qualities have longer game lengths and smaller score differences, both of which we believe are important in the application of accompanying game AI.

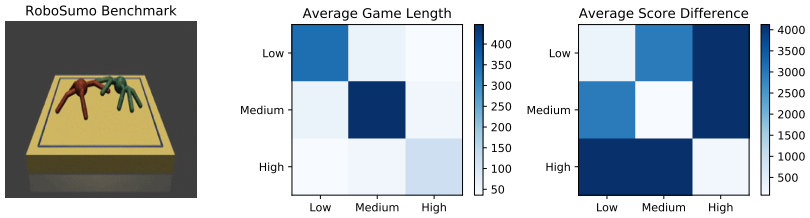


Figure 4: An illustration of the RoboSumo benchmark (left). The game length (middle) and the score difference (right) in an episode, where agents from different quality levels compete with each other.

Table 3: The QSD scores on RoboSumo tasks.

MuJoCo Tasks		PBT	DvD-TD3	QSD-PBT (Ours)
RoboSumo	score > 0%	1.20 ± 0.20	3.02 ± 0.10	$\mathbf{3.32} \pm \mathbf{0.02}$
R_{max}: 2k	score > 60%	0.30 ± 0.15	1.13 ± 0.02	$\mathbf{1.29} \pm \mathbf{0.02}$

B.2 Ablation Study

B.2.1 The adaptive diversity loss

We demonstrate the effects of adaptive diversity loss constrained by the population quality described in Section 3.2. The Atari game FishingDerby is used for the ablation study, and three training settings are compared: (1) The baseline setting is the diversity loss function with a fixed coefficient λ_∞. (2) In the decay setting, we apply exponential decay of the coefficient λ_∞ with decay step t_0. (3) In the adaptive setting, we apply an adaptive coefficient $\lambda_j(t, R_{t,j})$ that considers both the relative quality of the agent and the current training steps according to Section 3.2. The hyperparameters for this ablation study are detailed in Table 11.

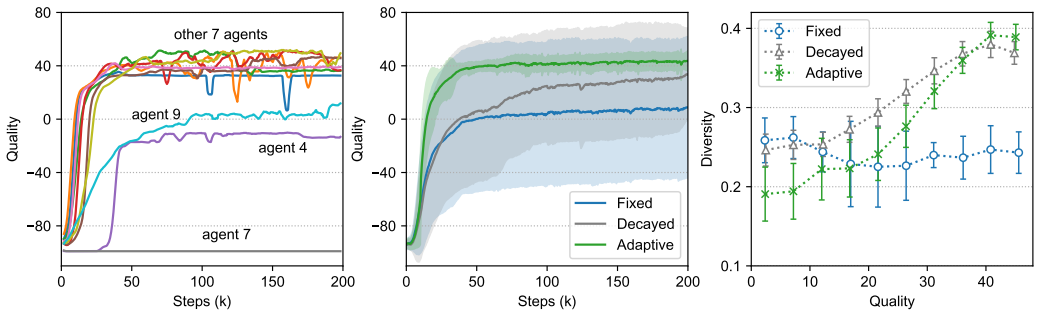


Figure 5: Comparison of three diversity training settings: the fixed diversity loss (fixed), the decayed diversity loss (decayed), and the adaptive diversity loss with quality constraint (adaptive). **Left**: typical training curves of the qualities of a population of 10 agents with the fixed setting. **Middle**: training curves of the qualities of a population of 10 agents for the three settings. **Right**: the QSD performance for the three settings.

The experimental results of the ablation study are shown in Figure 5. For the training setting of a fixed coefficient $\lambda^{(0)}$, it is worth noting that the qualities of some agents can get trapped into local optima, as shown in Figure 5 (left), which hinders the optimization of diversity among policies of similar qualities. Moreover, as shown in Figure 5 (middle), the qualities of agents improve slowly for both the fixed and the decay training settings, which may have a negative effect on maximizing diversity among policies of near-optimal qualities, e.g., quality levels above 40 in this game. By introducing the adaptive quality constraint, we observe a much faster improvement of the population quality in Figure 5 (middle). The overall QSD performances of the three settings are shown in Figure 5 (right). We can observe that the quality constraint optimization has slightly better QSD scores at

higher quality levels (>60%) although it improves the quality at the cost of diversity degradation at low quality levels. We recommend applying the adaptive diversity loss balanced by the population quality in practice.

We further investigate the effect of the hyperparameter decay rate t_0 and initial λ_0 in the QSD-PBT algorithm. In the Atari game FishingDerby, we first fix the initial λ_0 to 0.1 and vary the decay rate t_0 from 20k to 500k training steps, then fix the decay rate t_0 to 500k and vary the initial λ_0 from 0.05 to 0.4. Ablation results are shown in Figure 6, where computational/time overheads and QSD scores are measured in the same way as in Table 1 and Table 2. From the experimental results, we can conclude that the initial λ_0 significantly affects the training convergence and demonstrates a trade-off between the final QSD score and the training overhead in a large range (from 0.05 to 0.3). When λ_0 is too large ($\lambda_0 = 0.4$), QSD-PBT can not reach the optimal quality within an acceptable time overhead (500k steps), then the partial sum of the quality intervals is reported and thereby the QSD score degrades. Compared with λ_0, QSD-PBT algorithm is less sensitive to the decay rate t_0, higher decay rates (20 and 50) lead to similar results as lower initial λ_0, while a suitable decay rate (200) would both maintain the QSD score and reduce the time overheads.

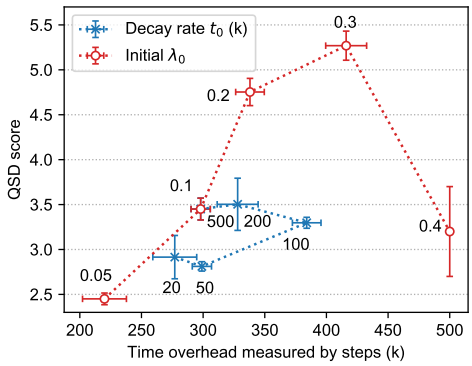


Figure 6: The final QSD scores and the time overheads under different settings of decay rate t_0 and initial λ_0.

B.2.2 Other Diversity Measure

In this section, we provide more analysis of the measure function. Note that the measure function is in need in our algorithm to measure the diversity among the policies, so that our algorithm can train to improve the diversity. In the main experiment, we choose the MSE function to be the measure function. We note that our method theoretically can be applied to any measure function as long as the measure function is differentiable. We provide one experiment for example below. However, the choice of the measure function absolutely will affect the performance of the algorithm, and we discuss this later.

We claim that QSD-PBT can be applied to any diversity measure as long as the diversity measure is differentiable with respect to the BDs. We here provide additional experimental results using a different (from the main experiments) diversity measure, i.e., the determinant of DPP, using the MuJoCo task Hopper.

Table 4: The QSD scores on the MuJoCo task Hopper using the determinant of DPP as the diversity measure. #step denotes the number of training steps.

MuJoCo Task		EDO-CS	PBT	DvD-TD3	QSD-PBT (Ours)
Hopper R_{max}: 3k	score > 0%	0.95 ± 0.43	0.80 ± 0.14	2.32 ± 0.96	$\mathbf{9.49 \pm 2.83}$
	score > 60%	0.10 ± 0.04	0.09 ± 0.03	0.31 ± 0.09	$\mathbf{0.45 \pm 0.12}$
	#step (k)	200 ± 27	$\mathbf{111 \pm 7}$	208 ± 27	170 ± 10

From the results presented in Table 4, we can conclude that QSD-PBT still outperforms other methods when the diversity measure is the determinant of DPP. The results are consistent with that in Table 1 when the pair-wise distance is employed as the diversity measure. One reason is that QSD-PBT optimizes the user-defined diversity objective directly, while the other methods optimize a task-agnostic diversity measure (PBT, DvD) or quality within different cells of the BD space (EDO-CS). Besides, QSD scores within intervals above 60% of the maximum qualities in Table 4 drop faster compared to the results on the pair-wise distance diversity measure in Table 1. The reason is that the determinant of DPP is much more sensitive to the changes in BDs, in comparison with the mean pair-wise distance measure. Therefore, we recommend the mean pair-wise Euclidean distance as the first choice for users and experiments in this paper.

The choice of measure function will have an impact on the training overheads from two aspects:

(1) The measure function itself may have high computational complexity, e.g., $O(n^3)$ in DPP, where n is the population size. However, n is small in practice (8 in MuJoCo and 10 in Atari) and the complexity is negligible compared to neural network models. For example, when we increase population size n from 10 to 100, we do not observe any decrease of GPU sample speed during training.

(2) Different measure functions will affect the convergence of QSD-PBT and hence affect the computational sources. In Figure 7, we provide training curves of MSE and DPP measure functions in MuJoCo tasks. From the figure, we can see the convergence rate of DPP is slightly slower than MSE. The reason is that MSE calculates the pairwise distance of BDs among all policies in the population and averages them by equal weights, while the DPP is more sensitive to the most similar policies and is unstable. For example, the DPP of a population that contains two same policies in it will be zero, regardless of how diverse the other policies are in the population.

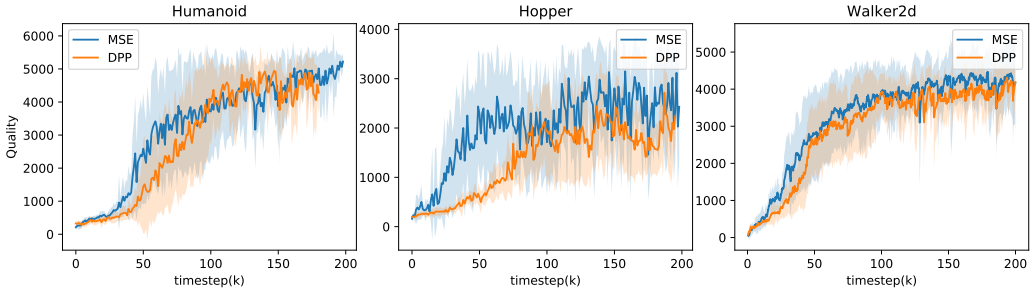


Figure 7: Comparison of the training curve between MSE and DPP measure function. Lines and shadings represent the mean and 1-std range of the qualities among the policies in the population.

B.3 Results on randomly designed BDs

Since the trajectory BD can be technically anything, the design principles in this paper are focused on generality and practicality. The BDs in previous experiments include various forms, e.g., action BD (MuJoCo joint torques), state BD (Atari game time), and trajectory BD (Atari left/right preference). However, we find that particular choices of BD still favor certain algorithms. For example, BD defined on actions favors the DvD algorithm since it directly optimizes the KL distance of agents' actions. Hence the improvement of DvD over PBT is more significant in the MuJoCo results (Figure 1) than in the Atari results (Figure 2). Instead of defining meaningful and explicit BDs, we further investigate the performance of QSD-PBT on implicit BDs that would less favor certain algorithms. Since we can only get states as raw pixel images from the Atari environment, we focus on actions and the implicit BD is designed as the randomly weighted action rate:

$$b_i(\tau) = \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{a}, \quad \boldsymbol{w}_i \sim \{(w_i^1, ...w_i^K) | \sum_{j=1}^{K} w_i^j = 1; w_i^j > 0, \forall j\} \tag{25}$$

where weight $\boldsymbol{w}_i$ is sampled from the normal distribution and normalized by the *softmax* function, $K = 6$, action rates $\boldsymbol{a}$ is a 6-dimension vector [*fire*, *noop*, *up*, *down*, *left*, *right*], the calculation of action rates is in Table 9. We generate five random BDs, i.e., the weights form a 5×6 matrix,

and provide additional training in the Atari game FishingDerby, the results are shown in Figure 8. Besides, QSD-PBT checkpoints trained on generally designed BDs in Section 4.2 are cross-evaluated by random BDs, marked as "CE". From the experiment, the results are consistent with Figure 1 and Figure 2. QSD-PBT CE is trained with the BD *fire rate* and the diversity of DvD-PPO is defined by action pairs, therefore both of them favor certain random BDs and perform slightly better than PPO and PBT. Comparing QSD-PBT and QSD-PBT CE, we conclude that our algorithm is capable of "what you will get is what you have defined" since it allows for the direct computation of the diversity gradient.

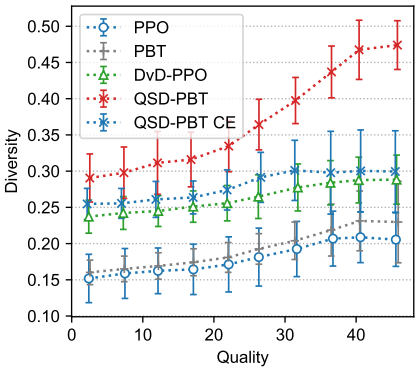


Figure 8: The quality similar diversity across 10 quality intervals on random BDs, "CE" means cross-evaluation on these BDs.

B.4 Diversity Results on Each Individual BD

We claim that QSD-PBT can generate user-intended diversity as long as a diversity measure function and a set of user-defined BDs are provided. We illustrate this ability of QSD-PBT in the main text using orientation BDs in Figure 3. We here provide additional results using the four Atari games for each of the five BDs defined for the Atari experiments.

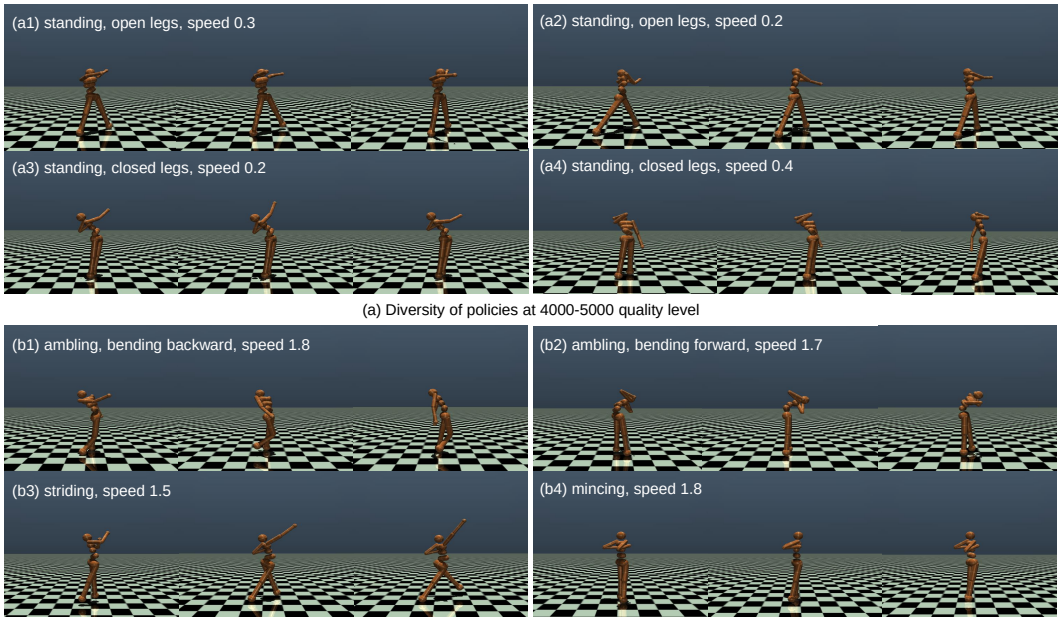
Results in Table 2 and Figure 2 are further decomposed into each individual BD in Figure 10, where we plot the QSD for a single BD one at a time, i.e., $\mathcal{B}(\pi_\theta) = [B_1(\pi_\theta)]$. After being normalized by scale factors in Table 9, the five BDs have similar ranges. Overall from Figure 10, we can conclude that QSD-PBT outperforms previous methods in terms of the QSD performance for each of the five BDs defined for the Atari experiments in most cases.

We observe that some user-defined BDs are highly correlated with the quality, e.g., the **game time** in PvE-style games (MsPacMan and RiverRaid), where QSD-PBT achieves only slight improvements compared with other methods. How to adaptively handle the correlation between a BD and the quality remains an interesting question. Besides, for the **left_right** BD in the game RiverRaid, the river course is narrow (shown in supplementary videos), and there is no shortcut (as in the game MsPacMan) that connects the left and right areas. Therefore, left and right actions are restricted to have similar numbers, and the resulting diversity scores are small for all the methods in this game.

B.5 Additional Visualizations

We illustrate the diversity at different quality levels in Figure 9 using the Humanoid-v2 task. In this task, an agent can be rewarded by standing or walking, and walking at faster speeds often leads to higher rewards. In Figure 9(a), agents at a lower quality level (4000-5000) learn to stand and balance in various poses, e.g., with open or closed legs and different hand gestures. In contrast, agents at a higher quality level (5000-6000) prefer various walking gaits, including ambling, striding, and mincing, which is demonstrated in Figure 9(b).

Additional visualizations of diverse policies generated by QSD-PBT are provided in Atari and MuJoCo videos. The results demonstrate that QSD-PBT is effective in both maximizing diversity



(a) Diversity of policies at 4000-5000 quality level

(b) Diversity of policies at 5000-6000 quality level

Figure 9: Visualization of diversity at different quality levels on Humanoid-v2.

across different quality levels and generating user-intended diversity across different user-defined BDs.

For the MuJoCo videos, we focus on demonstrating the diversity at different quality levels. For the Humanoid task, agents at a high quality level (5000-6000) prefer various walking gaits, e.g., ambling, striding, and mincing. Agents at a low quality level (4000-5000) only learn to stand still and balance in various poses. For the Walker2d task, agents at a high quality level (5000-6000) learn to run or stride, while agents at a low quality level (4000-5000) can only walk with small steps or just hop. For the Hopper task, agents at a high quality level (3000-4000) prefer fast hopping with different poses, while agents at a low quality level (2000-3000) only learn to stand or jump with small steps.

For the Atari videos, we focus on demonstrating the diversity of the five user-defined BDs. Aside from a video version of Figure 3 in MsPacMan, the orientation BDs (**left_right** and **up_down**) are further visualized in FishingDerby, where PPO agents always fish in the up and left area since it is an optimal and easy-to-explore solution. The diversity on the other three BDs (**game time**, **fire rate**, and **action continuity**) are demonstrated in IceHockey and RiverRaid. In the game IceHockey, agents get scores by various strategies, and three of them are displayed: (1) Dribble, run, and shoot (fire) in the normal style. (2) Stand still, defend, and shoot far away. (3) Swing left and right without shooting. In the game RiverRaid, agents can speed up or slow down by up or down actions, and therefore the total game time differs. Our agents can generate diverse behaviors by considering multiple aspects during driving, e.g., the speed, obstacles, and the remaining fuel.

B.6 Trade-off between quality and diversity

Generally speaking, the quality and the diversity are a trade-off, higher quality tends to result in lower diversity. But as shown in Figure 1, Figure 2 and Figure 6, since environments and BDs are so varied, it is hard to clarify the accurate correlation at each quality level and we need to analyze it case-by-case. Usually it depends on (1) the correlation of the BDs and the quality, and (2) the exploration space and complexity of the environment. Here we observe the difference between Figure 1 and Figure 2. In MuJoCo tasks, the QSD curve shows the quality and diversity trade-off, while in Atari games, the trade-off disappears.

We conclude with two major reasons: (1) The definition of BDs. For MuJoCo tasks, we define most of the BDs on joint torques, which are highly correlated to the pose control of the robots and the

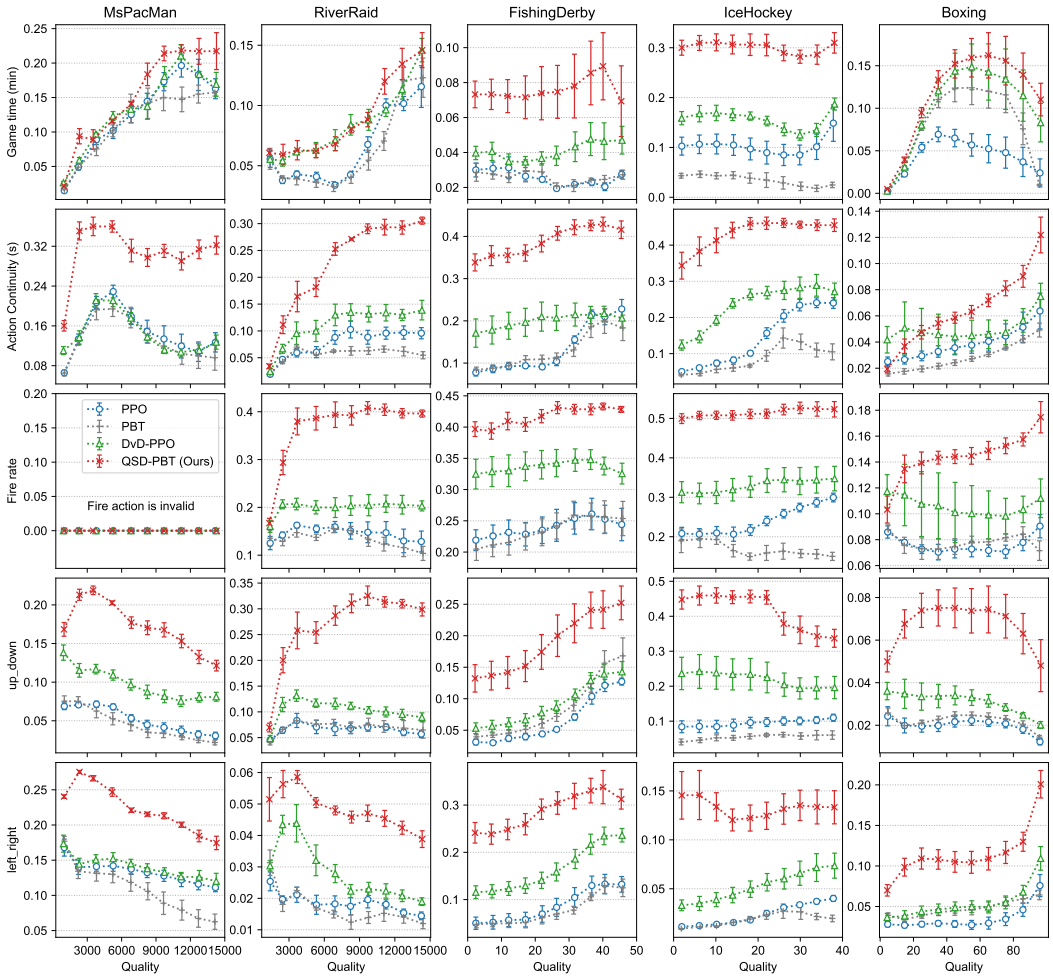


Figure 10: Detailed diversity results for individual BDs in Atari games

resulting score. While for Atari games, we define relatively coarse-grained BDs that have lower correlations with the quality, e.g., the left or right preference in Figure 3 does not influence the score since most games are mirror symmetric. (2) The exploration space and complexity of the environment. Compared with MuJoCo tasks, Atari games are more complex and have more exploration space. Take the game RiverRaid for example, at a lower quality level (<3000), the initial river course is narrow and the agent can only go straight. When it reaches certain scores, the agent will enter into new scenes that have a much broader course. The scene switching provides more exploration and diversity space, resulting in the improvement of diversity along with quality.

C Implementation Details

C.1 Descriptions of the BDs

For MuJoCo tasks, the BDs are defined on the scoring speed and the built-in joint torques respectively. The joint torques are the actions in MuJoCo tasks. The action spaces for each MuJoCo task are shown in Table 5-7, and more details can be found in https://www.gymlibrary.ml/environments/mujoco/. For a trajectory, we divide its total score by the number of frames used to obtain the BD of **Scoring speed**. For each joint in a trajectory, we sum the actions, i.e., the torques, applied to it and divide the sum by the number of frames to obtain a joint BD. As a result, the number of BDs for each MuJoCo task is the number of joints in that task plus one, i.e., the **Scoring speed**. We normalize each BD to

the range of $[0, 1.0]$. An overview of all the BDs for each task in the MuJoCo experiment is presented in Table 8,

Table 5: Action space for Hopper.

Num	Description (Torque applied to different joints)	Range
1	the thigh rotor	$[-1, 1]$
2	the leg rotor	
3	the foot rotor	

Table 6: Action space for Humanoid.

Num	Description (Torque applied to different joints)	Range
1	the hinge in the y-coordinate of the abdomen	$[-0.4, 0.4]$
2	the hinge in the z-coordinate of the abdomen	
3	the hinge in the x-coordinate of the abdomen	
4	the rotor between torso/abdomen and the right hip (x-coordinate)	$[-0.4, 0.4]$
5	the rotor between torso/abdomen and the right hip (z-coordinate)	
6	the rotor between torso/abdomen and the right hip (y-coordinate)	
7	the rotor between the right hip/thigh and the right shin	
8	the rotor between torso/abdomen and the left hip (x-coordinate)	
9	the rotor between torso/abdomen and the left hip (z-coordinate)	
10	the rotor between torso/abdomen and the left hip (y-coordinate)	
11	the rotor between the left hip/thigh and the left shin	
12	the rotor between the torso and right upper arm (coordinate -1)	
13	the rotor between the torso and right upper arm (coordinate -2)	
14	the rotor between the right upper arm and right lower arm	
15	the rotor between the torso and left upper arm (coordinate -1)	
16	the rotor between the torso and left upper arm (coordinate -2)	
17	the rotor between the left upper arm and left lower arm	

Table 7: Action space for Walker2d.

Num	Description (Torque applied to different joints)	Range
1	the right thigh rotor	$[-1, 1]$
2	the right leg rotor	
3	the right foot rotor	
4	the left thigh rotor	
5	the left leg rotor	
6	the left foot rotor	

For Atari games, we define a common set of BDs for each game. There are 5 different BDs in total. The design of these BDs are motivated by covering different ways in which diverse and meaningful policies could possibly differ. We normalize each BD to a similar range. An overview of all the BDs in the Atari experiment is presented Table 9.

C.2 The Diversity Measures

We use the mean pair-wise distance as the diversity measure for the main experimental results. For a population of policies Π with size N, $\Pi = \{\pi_{\theta_j} | 1 \leq j \leq N\}$, the mean pair-wise distance is defined

Table 8: An overview of all BDs for each MuJoCo task. #frame is the total number of frames in a trajectory.

BD	Definition	Normalization
Scoring speed	$\frac{\text{total_score}}{\#\text{frame}}$	$\frac{1}{5}x$
Joint torques in Hopper ($\text{dim} = 3$)	$\frac{\sum actions}{\#\text{frame}}$	$\frac{x+1}{2}$
Joint torques in Humanoid ($\text{dim} = 17$)	$\frac{\sum actions}{\#\text{frame}}$	$\frac{x+0.4}{0.8}$
Joint torques in Walker2d ($\text{dim} = 6$)	$\frac{\sum actions}{\#\text{frame}}$	$\frac{x+1}{2}$

Table 9: An overview of all BDs for Atari games. # indicates the total number of frames, action changes, or specific actions in a trajectory.

BD	Definition	Scale factor
Game time (minute)	$\frac{\#\text{frame}}{900}$	$\frac{1}{3}$
Fire rate (per second)	$\frac{15\cdot\#\text{fire}}{\#\text{frame}}$	$\frac{1}{3}$
Action continuity	$\log[\frac{15\cdot\#\text{action_change}}{\#\text{frame}}]$	$\frac{1}{15}$
Left_right	$\log[\frac{\#\text{left}}{\max(\#\text{right},1)}]$	$\frac{1}{15}$
Up_down	$\log[\frac{\#\text{up}}{\max(\#\text{down},1)}]$	$\frac{1}{15}$

as:

$$\frac{2}{N(N-1)}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}||\mathcal{B}(\pi_{\theta_i})-\mathcal{B}(\pi_{\theta_j})||_2, \tag{26}$$

where $\mathcal{B}(\pi_\theta)$ is the vector of all the BD values of a policy defined in Section 2.

As we stated in the main text, our method QSD-PBT can be applied to any explicit diversity measure as long as the measure is differentiable with respect to $\mathcal{B}(\pi_\theta)$. Later in Appendix B.2.2, we present results using another diversity measure, i.e., the determinant of a DPP. The DPP determinant of $\Pi = \{\pi_{\theta_j}|1 \leq j \leq N\}$ is defined as:

$$\det[\mathcal{K}(\mathcal{B}(\pi_{\theta_i}),\mathcal{B}(\pi_{\theta_j}))_{i,j=1}^{N}], \tag{27}$$

where $\mathcal{K}$ is a given kernel function. We set $\mathcal{K} = \exp[-\frac{||\mathcal{B}(\pi_{\theta_i})-\mathcal{B}(\pi_{\theta_j})||_1}{2}]$ in our case.

C.3 The Calculation of the QSD Score

In practice, the number of policies obtained throughout training with qualities lying in the same interval can be much larger than N, which is the population size of Π used for evaluating the diversity measure $\text{Div}(\Pi)$. To make an efficient and fair comparison across different methods in terms of the QSD score, for each quality interval we sample a population Π of size N for 100 times. The way we sample Π is by alternating the training index of the policy till N policies are obtained, because policies with the same training index tend to have similar BD values. The training index of a policy denotes which agent in the population the policy comes from during population-based training. We calculate the diversity measure for each sampled Π and use the sample mean of the 100 evaluations as the diversity measure for the corresponding quality interval, which is then aggregated according to Equation 1 to produce the QSD score.

C.4 Hyperparameters and Details of the Compared Baseline Methods

For MuJoCo tasks, we compare QSD-PBT to three population-based training algorithms: EDO-CS, PBT, and DvD. EDO-CS uses the ES for optimizing the quality and a selection mechanism to induce

diversity among the defined BDs. PBT uses TD3 as the backbone and mostly focuses on the quality of each policy in the population. The *Perturb* exploration and *Truncation selection* exploitation strategies are adopted every 10k training steps for our setting of PBT. The DvD-TD3 in the original paper is implemented. DvD-TD3 optimizes a combined loss of quality and a task-agnostic diversity measure defined on state action probabilities. All the hyperparameters for each method are listed in Table 10 except for EDO-CS and QD-PG, for which we implement with the architecture and hyperparameters suggested in their papers. For QSD-PBT, we use the LSTM as a feature exactor in BD estimators. We pass the trajectory state and trajectory action to an LSTM, respectively, then concatenate the feature and pass it to the MLP. The 'state' BD we used for reproducing QD-PG is the action (a real-valued vector, which has a dimension of 17, 3, and 6 respectively in the three MuJoCo tasks) in a state. We believe this is a fair setting for QD-PG, because the trajectory BD we defined here is the average torques (i.e., actions) applied to the hinge joints over a trajectory.

Table 10: Hyperparameters for MuJoCo tasks.

Hyperparameters	Value
Shared	
Population size (N)	8
Replay buffer size	$10^6 \times N$
Mini-batch size	$1024 \times N$
Optimizer	Adam
Learning rate	3e-4
Discount factor (γ)	0.99
Target network update rate (τ)	0.005
Std of gaussian exploration noise	0.1
Delayed policy update	2
Policy noise	0.1
Noise clip	0.5
Policy network	MLP (state_dim-256-256-action_dim)
Critic network	MLP (state_dim+action_dim-256-256-1)
BD Estimator	LSTM(128) + MLP (256-256-BD_dim)
PBT	
Exploitation interval	every 10k steps
Truncation selection ratio	0.25
DvD-TD3	
DvD loss factor	0.5
DvD embedding mini-batch size	256
QSD-PBT	
Adaptive diversity loss $\lambda^{(0)}$	0.1 (Hopper, Humanoid), 0.5 (Walker2d)
Adaptive diversity loss t_0	2×10^4

For Atari games, we replace the baseline method EDO-CS with PPO for training efficiency consideration, where a population of N independent PPO agents are trained. As Atari games have discrete actions and high-dimensional image inputs, all the methods employ the DQN (Mnih et al., 2015) architecture as the backbone model. We find it effective to provide statistics from the beginning of a game to the current state (the number of frames, the number of fire, left, right, up, down actions and the number of action changes) as a sufficient encoding of trajectory $\tau_{0:j}$. Instead of the LSTM applied in MuJoCo experiments, these additional features are applied in the DQN model to better estimate BD defined in Table 9. Accordingly, we use PPO in the implementation of both DvD and PBT. All the hyperparameters for each method are listed in Table 11.

C.5 Pseudocode of QSD-PBT

The developed QSD-PBT is in general a population-based RL algorithm. In QSD-PBT, we employ parallel actors and learners to simultaneously train a population of N agents. The policy of each

Table 11: Hyperparameters for Atari games.

Hyperparameters	Value
Shared (PPO)	
Population size (N)	10
Number of actors	700 (CPUs)
Number of learners	8 (GPUs)
Frame skip	4
Reward shaping	$\log(\text{abs}(r)+1)\cdot(2\cdot\mathbf{1}_{\{r\geq 0\}}-\mathbf{1}_{\{r<0\}})(Fanet\ al.,\ 2021)$
Replay buffer size	$50000\times N$
Mini-batch size	$256\times N$
Optimizer	Adam
Learning rate	1e-4
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
PPO clipping ratio	0.2
Value function coefficient c_1	0.5
PBT	
Exploitation interval	every 20k steps
Truncation selection ratio	0.2
DvD-PPO	
DvD loss factor	0.1
DvD embedding mini-batch size	512
QSD-PBT	
Adaptive diversity loss $\lambda^{(0)}$	0.1
Adaptive diversity loss t_0	5×10^5

agent is trained using the gradient of the loss function defined in Equation 9. Meanwhile, the policy value functions or Q functions are trained using the mean squared error, and so are the state BD estimators or the state-action BD estimators of each agent. Moreover, QSD-PBT maintains a running average estimation of each agent's quality, each agent's BDs, and the population's mean quality. The pseudocode of QSD-PBT is given in Algorithm 1. Note that, without loss of clarity, we present QSD-PBT with TD3 and QSD-PBT with PPO in one algorithm.

Algorithm 1: QSD-PBT with TD3 or PPO baseline

Initialize network parameters for a population of N agents: $\{\theta_1, \theta_2, ..., \theta_N\}$.
Start multiple actors and learners in parallel.
Actors: **while** *true* **do**
- Fetch the latest model from the learners, and run multiple game episodes.
- Generate RL training samples in the form: $x = [a_t, s_t, s_{t+1}, r_t]$, and the encoding result of $\tau_{0:t-1}$, and send them to the replay buffer.
- For each agent, calculate the average BD results $\hat{\mathcal{B}}(\pi_\theta)$ and the quality R using results in multiple games, and send R, $\mathcal{B}(\pi_\theta)$, $\hat{\mathcal{B}}(\pi_\theta)$ to a monitor.

Learners:
for $t \in 1, 2, 3, ...$ **do**
- Fetch current BD results $\mathcal{B}(\pi_\theta)$, $\hat{\mathcal{B}}(\pi_\theta)$, and qualities R of all agents from the monitor.
- Calculate the diversity measure function $\mathrm{Div}(\Pi)$.
- **for** $n \in 1, 2, ..., N$ **do**
  - Calculate the diversity gradient of $\mathrm{Div}(\Pi)$ at $\hat{\mathcal{B}}$.
  - Fetch a batch of samples for agent n from the replay buffer.
  - Calculate value loss and policy loss according to TD3 or PPO algorithm.
  - Calculate mean-square loss for BD estimators $V_B^{\pi_{\theta_n}}$ by $\mathcal{B}(\pi_{\theta_n})$.
  - Calculate diversity loss according to Equations 6, 8.
  - Update critic parameters θ_n using gradients on value loss and BD estimator loss.
  - Update actor parameters θ_n using gradients on $L_{total}(\pi_{\theta_n})$ as in Equation 9.