SAMPLE EFFICIENT QUALITY DIVERSITY FOR NEURAL CONTINUOUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel Deep Neuroevolution algorithm, QD-RL, that combines the strengths of off-policy reinforcement learning (RL) algorithms and Quality Diversity (QD) approaches to solve continuous control problems with neural controllers. The QD part contributes structural biases by decoupling the search for diversity from the search for high return, resulting in efficient management of the exploration-exploitation trade-off. The RL part contributes sample efficiency by relying on off-policy gradient-based updates of the agents. More precisely, we train a population of off-policy deep RL agents to simultaneously maximize diversity within the population and the return of each individual agent. QD-RL selects agents interchangeably from a Pareto front or from a Map-Elites grid, resulting in stable and efficient population updates. Our experiments on the ANT-MAZE and ANT-TRAP environments show that QD-RL can solve challenging exploration and control problems with deceptive rewards while being over 15 times more sample efficient than its evolutionary counterparts.

1 INTRODUCTION

Natural evolution has the fascinating ability to produce organisms that are all high-performing in their respective niche. This peculiarity of open-ended evolution, which produced a tremendous diversity of living systems within one run, inspired the development of a new optimization paradigm. Quality-Diversity (QD) is a family of optimization algorithms that aim at searching for a collection of both diverse and high-performing solutions. While classic optimization methods focus on finding a single efficient solution, the role of QD optimization is to cover the range of possible solution types and to return the best solution for each type. This process is sometimes referred to as "illumination" in opposition to optimization, as the goal of these algorithms is to reveal (or illuminate) a search space of interest.

The QD literature generally builds on black-box optimization approaches such as evolutionary algorithms to optimize a population of solutions. These algorithms often rely on random mutations and/or crossover to explore small search spaces but struggle when confronted to higher-dimensional problems. As a result, QD methods often scale poorly to large sequential decision problems with continuous state and action spaces, where using controllers with many parameters such as deep neural networks is mandatory. Besides, evolutionary methods are very valuable when nothing is known about the function to optimize. But when facing a policy search problem that can be formalized as a Markov Decision Process (MDP), by using Policy Gradient (PG), one can exploit the analytical structure of neural networks to more efficiently estimate the gradient of the performance with respect to the policy parameters. It would therefore be a pity not to exploit these properties when the Markov assumption holds and when the controller is a neural network.

In this work, we aim to strongly improve the sample efficiency of QD methods to get neural controllers solving continuous action space MDPs. We build on off-policy PG methods to propose a new mutation operator that takes into account the Markovian nature of the problem being addressed and analytically exploits the known structure of the neural controller. Our QD-RL algorithm falls within the QD framework described by Cully & Demiris (2017) and takes advantage of its powerful exploration capabilities, but also demonstrates remarkable sample efficiency brought by off-policy RL methods. We compare QD-RL to several recent algorithms that also combine a diversity objective and a return maximization method, namely the NS-ES family (Conti et al., 2018) and the ME-ES

algorithm (Colas et al., 2020) and show that QD-RL is an order of magnitude more sample efficient than these competitors.

2 Problem Statement

We consider the general context of a fully observable Markov Decision Problem (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \rho_0)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, γ is a discount factor and ρ_0 is the initial state distribution. We aim to find a set of parameters θ of a parameterized policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ so as to maximize the objective function $\mathcal{J}(\theta) = \mathbb{E}_\tau \sum_t \gamma^t r_t$ where τ is a trajectory obtained from π_θ starting from state $s_0 \sim \rho_0$ and r_t is the reward obtained along this trajectory at time t. We define the Q-value for policy π, $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ as $Q^\pi(s, a) = \mathbb{E}_\tau \sum_t \gamma^t r_t$, where τ is a trajectory obtained from π_θ starting from s and performing initial action a.

QD aims at evolving a set of solutions θ that are both diverse and high performing. The measure of diversity relies on the definition of a Behavior Descriptor (BD) space, which is used to characterize solutions in functional terms, in addition to their score $\mathcal{J}(\theta)$. We note bd_θ the BD of a solution θ. The BD space is often designed using relevant features of the task goal. For instance, in robot navigation tasks, a pertinent BD is the final position of the robot at the end of a trajectory. In robot locomotion, it may rather be the position and velocity of the robot center of gravity at specific times. BDs are used to define the notion of novelty, i.e., a solution is novel if its behavior descriptors are different, in a chosen metric, from the ones of the solutions obtained so far.

In this work, we define an additional notion that we call *state Behavior Descriptor*. A state BD, noted bd_t, is a set of features extracted from a state $s_t \in \mathcal{S}$ at time step t that are relevant to the task. From state BDs, we define the BD of a solution θ as a function of all state BDs encountered by policy π_θ when interacting with the environment, as illustrated in Figure 1. More formally, we note $bd_\theta = \mathbb{E}_\tau \left[f_{bd}(\{bd_1, \ldots, bd_T\}) \right]$, where T is the trajectory length and f_{bd} is an aggregation function. For instance, f_{bd} can average over state BDs or return only the last state BD of the trajectory. If we consider again robot navigation, a state BD bd_t may represent the position of the robot at time t and the solution BD bd_θ may be the position of the robot at the end of the trajectory. State BDs are used to measure the novelty of a state relatively to all states encountered so far.

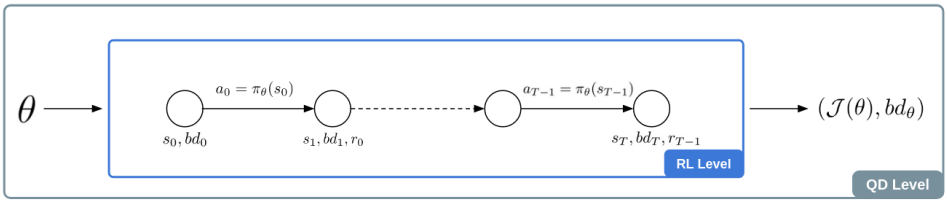


Figure 1: The RL part of QD-RL operates at the time step level while its QD part operates at the controller level, considering the MDP as a black box. The controller score is defined as the discounted sum of the time steps rewards and its BD is computed as an aggregation function function over all the state BDs along the trajectory.

3 Related work

A distinguishing feature of our approach is that we combine diversity seeking at the level of trajectories using solution BDs bd_θ and diversity seeking in the state space using state BDs bd_t. The former is used to select agents from the archive in the QD part of the architecture, whereas the latter is used in applying policy gradient steps in the RL part, see Figure 2. We organize the literature review below according to this split between two types of diversity seeking mechanisms.

With this organization, we have to put aside some families of methods that are related to our work in a lesser extent, such as algorithms combining evolutionary approaches and deep RL Pourchot & Sigaud (2018); Khadka & Tumer (2018); Khadka et al. (2019), algorithms maintaining a population of RL agents for exploration without an explicit diversity criterion (Jaderberg et al., 2017) or algo-

rithms explicitly looking for diversity but in the action space rather than in the state space (Doan et al., 2019; Jung et al., 2020; Parker-Holder et al., 2020).

Seeking for diversity and performance in the space of solutions Simultaneously maximizing diversity and performance is the central goal of Quality-Diversity methods (Pugh et al., 2016; Cully & Demiris, 2017). Among the various possible combinations offered by the QD framework, Novelty Search with Local Competition (NSLC) (Lehman & Stanley, 2011b) and MAP-Elites (ME) (Mouret & Clune, 2015) are the two most popular algorithms. In ME, the BD space is discretized into a grid used to store solutions. Each cell of the grid is a niche of the BD space and the selection mechanism samples individuals uniformly from all cells. On the other hand, NSLC builds on the Novelty Search (NS) algorithm (Lehman & Stanley, 2011a) and maintains an unstructured archive of solutions selected for their local performance. Cully & Mouret (2013) augment the NSLC archive by replacing solutions when they outperform the already stored ones. In QD-RL, we build on the standard ME approach and on an augmented version of the NSLC algorithm where the population is selected from a global quality-diversity Pareto front inspired from Cully & Demiris (2017). Relying on these components, algorithms such as QD-ES and NSR-ES have been applied to challenging continuous control environments in Conti et al. (2018). But, as outlined in Colas et al. (2020), these approaches are not sample efficient and the diversity and environment reward functions could be mixed in a more efficient way. In that respect, the most closely related work w.r.t. ours is Colas et al. (2020). The ME-ES algorithm also optimizes both diversity and reward efficiency, using an archive, two ES populations and the MAP-ELITES approach. Using such distributional ES methods has been shown to be critically more efficient than population-based GA algorithms (Salimans et al., 2017), but our results show that they are still less sample efficient than off-policy deep RL methods as they do not leverage direct access to the policy gradient.

Finally, similarly to us, the authors of Shi et al. (2020) try to combine novelty search and deep RL by defining behavior descriptors and using an NS part on top of a deep RL part in their architecture. In contrast with our work, the transfer from novel behaviors to reward efficient behaviors is obtained through goal-conditioned policies, where the RL part uses goals corresponding to outcomes found by the most novel agents in the population. Most importantly, the deep RL part contains no diversity seeking mechanism in the state space.

Seeking for diversity and performance in the state space Seeking for diversity in the space of states or actions is generally framed into the RL framework. An exception is Stanton & Clune (2016) who define a notion of *intra-life novelty* that is similar to our *state novelty* described in Section 2. But their novelty relies on skills rather than states, and is implemented specifically for navigation tasks, whereas ours is more general. Our work is also related to algorithms using RL mechanisms to search for diversity only, such as Eysenbach et al. (2018); Pong et al. (2019); Lee et al. (2019); Islam et al. (2019). These methods have proven useful in the sparse reward case, but they are inherently limited when using the reward signal can facilitate exploration, as they ignore it. Other works sequentially combine diversity seeking and RL. The GEP-PG algorithm Colas et al. (2018) combines a diversity seeking component, namely *Goal Exploration Processes* (Forestier et al., 2017) and the DDPG deep RL algorithm (Lillicrap et al., 2015). This sequential combination of exploration-then-exploitation is also present in GO-EXPLORE (Ecoffet et al., 2019) and in PBCS (Matheron et al., 2020). Again, this approach is limited when the reward signal can help driving the exploration process towards a satisfactory solution. These sequential approaches first look for diversity in the space of trajectories, then optimize performance in the state action space, whereas we do so simultaneously in the space of trajectories and in the state space.

To the best of our knowledge, QD-RL is the first algorithm optimizing both diversity and performance in the solution and in the state space, using a sample efficient deep RL method for the latter.

4 METHODS

QD-RL evolves a population of N neural policies $(\theta_1, \dots, \theta_N)$ so as to maximise their expected return as well as the diversity within the population. It relies on an archive storing all candidate solutions for selection and on a shared replay buffer storing all transitions (s_t, a_t, r_t, bd_t) collected when training all agents. As illustrated in Figure 2, one iteration consists of three successive phases: population selection among the solutions in the archive, mutation of the population, and evaluation

of the newly obtained solutions. These phases are repeated until a maximum computational budget is reached or until convergence.

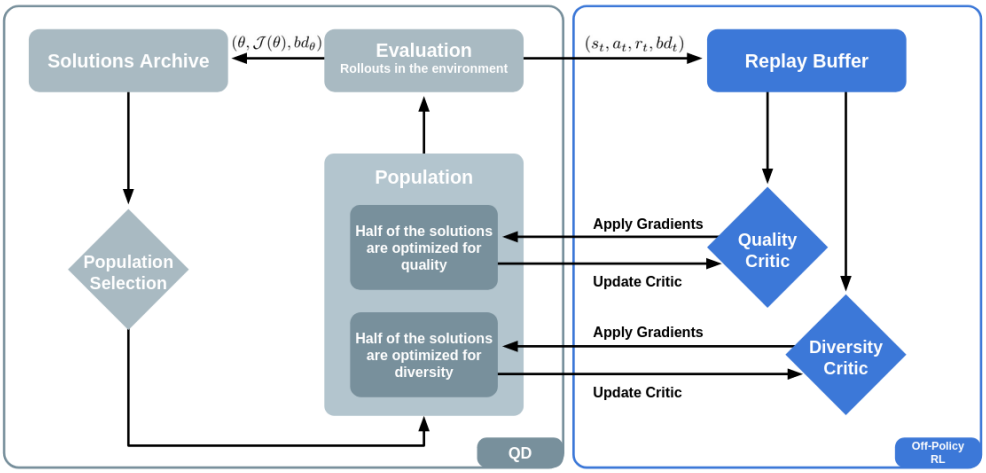


Figure 2: One QD-RL iteration consists of three phases: 1) A new population of solutions is sampled from the QD Pareto Front or from the Map-Elites grid. 2) These solutions are mutated by an off-policy RL agent: half of the solutions are optimized for quality and the other half for diversity. The deep RL agent leverages two shared critics, one for each objective. 3) The newly obtained solutions are evaluated by running the corresponding controllers in the environment. Transitions are stored in a replay buffer while final scores and BDs of solutions are stored in the QD archive.

Solution archive and selection mechanism. In this study, we consider two different QD configurations for archive management and selection mechanism. While they rely on different assumptions, we show in Section 5 that QD-RL manages to solve the tasks efficiently in both cases.

In the first setting, we maintain an unstructured archive of solutions θ found so far as well as their behavior descriptors bd_θ. The archive is a limited size FIFO list. Furthermore, we implement insertion rules inspired from Cully & Mouret (2013) to prevent adding solutions that are too close in the BD space unless they offer better performance (see Appendix B for more details). To select solutions from the archive, we compute a QD Pareto front of the solutions and sample the new population from this front. We use the $\mathcal{J}(\theta)$ objective function as the quality criterion, the diversity criterion is computed as the average distance between bd_θ and its nearest neighbors in the archive.

In the ME setting, the behavior descriptor space is discretized into a Cartesian grid that constitutes the ME archive. We use the standard ME insertion criterion: when adding a solution θ into the grid, we compute its BDs and find the corresponding cell. If the cell is empty, we insert the solution. Otherwise, we add the solution only if its quality score is greater than the one of the contained solution. To select a new population, we sample solutions uniformly from the grid.

Population mutation. During one iteration, we mutate each solution in the population to optimize its quality with respect to the objective function, but also to optimize the population diversity. To decouple exploration from exploitation, we introduce two mutation operators optimizing respectively for quality and for diversity. Both operators rely on a deep RL off-policy algorithm to compute gradients with respect to their respective criterion so as to to guide the mutation towards a promising direction. While any off-policy algorithm supporting continuous action could be used, we rely on Twin Delayed Deep Deterministic Policy Gradient TD3 (Fujimoto et al., 2018). We compute the novelty of a state s_t encountered by a policy at time step t as the mean distance between this state's bd_t and its nearest neighbors among BDs encountered so far. To this end, we keep a record of all collected BDs and filter them to get rid of the too similar ones. This novelty score is noted r_t^D. To homogenize notations, we note the environment reward r_t^Q. We introduce two critic networks Q_w^Q and Q_w^D, parameterized by vector w, that are shared by the population and used by TD3 to optimize solutions respectively for quality and diversity.

At each iteration, QD-RL mutates half of the solutions in the population to maximise quality and the other half to maximise diversity. As Colas et al. (2020), we use a 0.5 ratio not to bias the algorithm

towards exploration or exploitation. Nonetheless, we found that in practice QD-RL balances both criteria during training to solve the tasks. We leave for further study the automatic adjustment of this ratio.

To mutate a solution θ, we sample a batch of transitions from the replay buffer and use stochastic gradient descent (SGD) where the gradient is computed as the TD3 policy gradient, see Equation (1). We respectively use the reward r_t^Q when optimizing for quality, the reward r_t^D when optimizing for diversity, and their corresponding critics. Note that while the r_t^Q value is the one observed in the environment, the r_t^D value changes across training time as new states are encountered. To deal with this non-stationary reward, we recompute a fresh score r_t^D from the observed bd_t every time a batch is sampled. The gradient computation over solutions is performed in parallel. Every time a policy gradient is computed, we also update the critics Q_w^Q and Q_w^D. The critic gradients are computed in parallel and then averaged across the population. The global population update can be written as

$$\begin{aligned}
&\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} \sum_{batch} Q_w^Q(s_t, \pi_{\theta_i}(s_t)), \quad \forall i \leq N/2 \\
&\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} \sum_{batch} Q_w^D(s_t, \pi_{\theta_i}(s_t)), \quad \forall i > N/2 \\
&w \leftarrow w - \tfrac{2\alpha}{N} \nabla_w \sum_{batch} \sum_{i=1}^{N/2} \Big(Q_w^Q(s_t, a_t) - (r_t^Q + \gamma Q_{w'}^Q(s_{t+1}, \pi_{\theta_i}(s_{t+1}))) \Big)^2 \\
&w \leftarrow w - \tfrac{2\alpha}{N} \nabla_w \sum_{batch} \sum_{i=N/2+1}^{N} \big(Q_w^D(s_t, a_t) - (r_t^D + \gamma Q_{w'}^D(s_{t+1}, \pi_{\theta_i}(s_{t+1}))) \big)^2
\end{aligned} \tag{1}$$

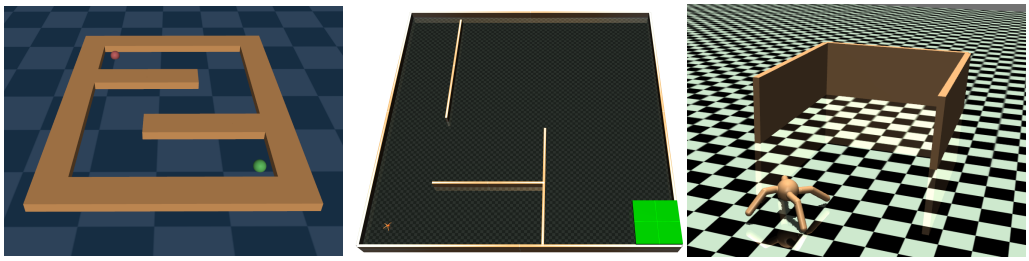
where w' are parameters of target critic networks. To keep notations simple, updates of the extra critic networks introduced in TD3, which serve to reduce the value estimation bias, do not appear in (1) although we use them in practice.

Evaluation. Once mutated, solutions are evaluated. To do so, each new solution θ plays trajectories with the resulting controller π_θ in the environment. All encountered transitions are stored in the replay buffer and we use the score $\mathcal{J}(\theta)$ and behavior descriptor bd_θ to place new solutions in the unstructured archive or in the MAP-Elites grid.

A thorough description of the QD-RL algorithm is available in Appendix D, Algorithm 1.

5 EXPERIMENTS

In this section, we demonstrate the capability of QD-RL to solve challenging exploration problems. We implement it with TD3 and refer to this configuration as the QD-TD3 algorithm. Hyperparameters are described in Appendix B. We first analyse each component of QD-TD3 and demonstrate their usefulness on a simple example. Then, we show that QD-TD3 solves a more challenging control and exploration problem such as navigating the MuJoCo Ant in a large maze with a better sample complexity than its evolutionary competitors. Finally, we demonstrate that QD-TD3 can solve control problems when the behavior descriptor space is not aligned with the task.



(a) POINT-MAZE (b) ANT-MAZE (c) ANT-TRAP

Figure 3: Experimental environments. Though the first two may look similar, the POINT-MAZE environment state and action spaces are two-dimensional, whereas in ANT-MAZE, the state space has 29 dimensions and the action space 8.

5.1 POINT-MAZE AND POINT-MAZE-INERTIA: MOVE A POINT IN A MAZE

We first consider the POINT-MAZE environment in which a 2D material point agent must exit from the maze depicted in Figure 3a. In this environment, BDs are defined as the agent coordinates (x_t, y_t) at time t and an action corresponds to position increments along the x and y axis.

In order to exit the maze, the agent must find the right balance between exploitation and exploration, that is, at a certain point to ignore the policy gradient and only explore the maze. Although this environment may look simple due to its low dimensionality, it remains very challenging for standard deep RL agents such as TD3, see Figure 5b. We also consider an identical task, POINT-MAZE-INERTIA, in which the point has momentum. In this case, behavior descriptors are unchanged but the observation contains velocities in addition to positions. An action corresponds to a force (F_x, F_y). Despite the additional complexity, we observe equivalent performance.

We leverage this environment to perform an ablation study so as to validate each component of our algorithm. First, we measure performance when maximizing quality only. We call the resulting agent Q-TD3, but this is simply a multi-actor TD3.

Then, we evaluate the agent performance when looking for solutions to maximize diversity only. We call the resulting agent D-TD3. Finally, we consider a D-TD3 + PARETO algorithm optimizing only for diversity but performing agent selection from the archive with a Pareto front. That is, it selects solutions for the next generation based on both their quality and diversity, but without optimizing the former. Table 1 summarises all these ablations.

Table 1: Summary of compared ablations on POINT-MAZE

	Opt. Quality	**Opt. Diversity**	**Selection mechanism**	**Episode return**
QD-TD3 + ME	✓	✓	MAP-ELITES	$\mathbf{-24}$ (± 0)
QD-TD3 + PARETO	✓	✓	PARETO	-27 (± 1)
D-TD3 + PARETO	X	✓	PARETO	-37 (± 3)
D-TD3	X	✓	None	-111 (± 53)
Q-TD3 + PARETO	✓	X	PARETO	-128 (± 0)
Q-TD3	✓	X	None	-130 (± 2)

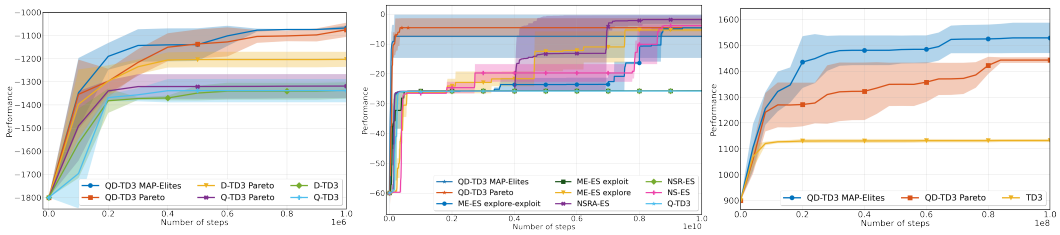
5.2 ANT-MAZE: CONTROL AN ARTICULATED ANT TO SOLVE A MAZE

ANT-MAZE is a challenging environment modified from OpenAI Gym (Brockman et al., 2016) based on ANT-V2 and also used in Colas et al. (2020); Frans et al. (2018). In ANT-MAZE, a four-legged ant robot has to reach a goal zone located in the lower right part of the maze (colored in green in Figure 3b). The initial position of the ant is sampled from a small circle around the initial point located in the extreme bottom left of the maze. Maze walls are organized so that following the gradient of distance to the goal drives the ant into a dead-end. As in POINT-MAZE, the reward is expressed as minus the distance between the center of gravity of the ant and the center of the goal zone, thus leading to a strongly deceptive gradient. Since the agent must learn to control a body with 8 degrees of freedom in all directions to explore the maze and solve it, it is higher-dimensional than POINT-MAZE and more complex than the standard ANT-V2 gym environment where the ant only learns to go forward. The observation space contains the positions, angles, velocities and angular velocities of most ant articulations and center of gravity, and has 29 dimensions. An action corresponds to the choice of 8 continuous torque intensities to apply to the 8 ant articulations. The episodes last 3000 time steps. As in POINT-MAZE, the outcome is computed as the final position (x_f, y_f) of the center of gravity of the agent.

On this benchmark, we compare the performance of QD-TD3 to four state-of-the art evolutionary methods: NSR-ES, NSRA-ES, NS-ES and ME-ES. While NS-ES and ME-ES-EXPLORE optimize only for diversity, and ME-ES-EXPLOIT optimizes only for quality, NSR-ES, NSRA-ES and ME-ES-EXPLORE- EXPLOIT optimize for both. To ensure fair comparison, we did not implement our own versions of these algorithms but reused results from the ME-ES paper (Colas et al., 2020). We also ensured that the environment we used was rigorously the same.

Table 2: Summary of compared algorithms. Performance is computed as the minimum distance to the goal reached at the end of an episode by the population.

	Final Perf. ($\pm$ std)	**Steps to -5**	**Ratio to** QD-TD3
QD-TD3 PARETO	-4 (± 3)	4e8	1
QD-TD3 ME	-7 (± 7)	4e8	1
NS-ES	-4 (± 1)	8.5e9	21
NSR-ES	-26 (± 0)	∞	∞
NSRA-ES	-2 (± 1)	7e9	17
ME-ES- EE	-5 (± 1)	8.5e9	21
ME-ES- EXPLORE	-6 (± 6.9)	7.5e9	19
ME-ES- EXPLOIT	-26 (± 0)	∞	∞
Q-TD3	-26 (± 0)	∞	∞



(a) POINT-MAZE-INERTIA (b) ANT-MAZE (c) ANT-TRAP

Figure 4: Learning curves of QD-TD3 versus ablations on POINT-MAZE and several other agents on ANT-MAZE and POINT-MAZE-INERTIA. The performance corresponds to the highest return on POINT-MAZE and to the highest negative distance to the goal area on ANT-MAZE. In Figures a) and c), all experiments are averaged over five seeds. In Figure b), we represented the two best seeds out of five for all algorithms.

5.3 ANT-TRAP: CONTROL AN ARTICULATED ANT TO AVOID A TRAP

Similarly to ANT-MAZE, the ANT-TRAP environment is derived from ANT-V2 and inspired from Colas et al. (2020). In ANT-TRAP, the four-legged ant initially appears in front of a trap and must avoid it in order to run as fast as possible in the forward direction along the x-axis. The trap consists of three walls that form a dead-end directly in front of the ant, leading to a strongly deceptive gradient. As in previous environments, the behavior space is defined as the final position (x_f, y_f) of the center of gravity of the agent. The peculiarity of this environment compared to the others is that here, the reward is expressed as the ant's forward velocity and is thus not fully aligned with the BDs. We can thus test the coverage capacity of our method in the case of a partial alignment. The action and observation spaces are the same as in ANT-MAZE and the episodes last 1000 time steps.

6 DISCUSSION

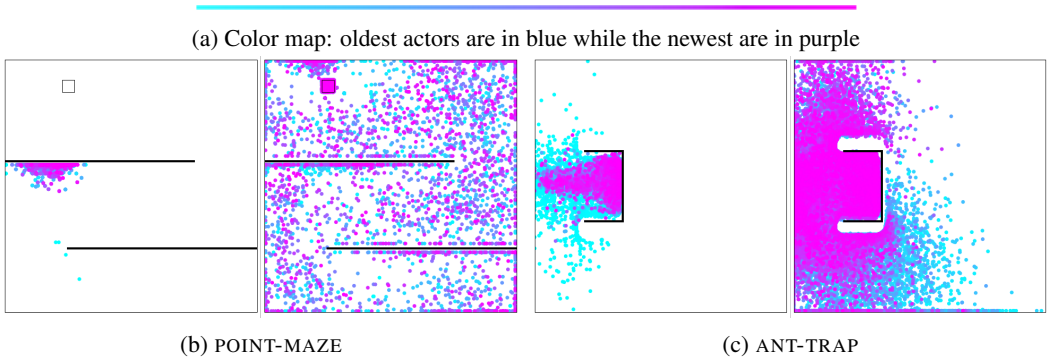
In this section, we discuss the performance of QD-RL and explain why it is more sample efficient.

Ablation study in POINT-MAZE. The ablation study in Table 1 shows that: 1) When maximising quality only, Q-TD3 with or without a selection mechanism fails due to the deceptive nature of the reward. 2) When maximising diversity only, D-TD3 fails without selection mechanism but adding Pareto selection is enough to exit the maze. 3) When optimizing quality and diversity, the agent finds a more optimal trajectory to the maze exit, resulting in higher returns. 4) While ME has slightly less variance, both QD methods solve the problem with high returns. This study validates the usefulness of the QD-RL components: 1) optimizing for diversity is required to overcome the deceptive nature of the reward; 2) a proper population selection mechanism is required to ensure proper convergence, and adding quality optimization provides better asymptotic performance.

Comparison to neuroevolution competitors in ANT-MAZE. Table 2 compares QD in terms of sample efficiency to Deep Neuroevolution algorithms in ANT-MAZE. QD-RL is run on 10 CPU cores

during 2 days while its competitors used several hundreds of CPU cores for the same duration. Nonetheless, QD-RL matches the asymptotic performance of ME-ES using $15\times$ less samples, thus explaining the lower resource requirements. We see three reasons for the improved sample efficiency of QD-RL: 1) QD-RL leverages a replay buffer and can re-use each sample several times. 2) QD-RL leverages a notion of novelty defined at the state level and thus can exploit all collected transitions to maximize both quality and diversity. For example, in ANT-MAZE, one trajectory collected brings 3000 samples to QD-RL while standard QD methods would consider it as a unique sample. 3) RL exploits the analytical gradient between the neural network weights and the resulting policy action distribution and thus estimates only the impact of the distribution on the return, whereas standard QD methods estimate directly the impact of the weights onto the return.

Performance assessment when BDs are not fully aligned with the task in ANT-TRAP. Finally, we evaluated QD-RL in the ANT-TRAP environment. The difference from previous environments is that reward is computed as the ant forward velocity at time t and not as an Euclidean distance to a target. As the BD remains the ant center of gravity, it is not directly aligned with the task. Nonetheless, we observe in Figure 5c that optimizing only for quality results in sub-optimal performance while QD-RL finds the optimal strategy by running around the trap and beyond.



(a) Color map: oldest actors are in blue while the newest are in purple

(b) POINT-MAZE

(c) ANT-TRAP

Figure 5: Coverage maps of Q-TD3 and QD-TD3 (respectively left and right) on both POINT-MAZE and ANT-TRAP environments.

7 CONCLUSION

In this paper, we proposed a novel way to deal with the exploration-exploitation trade-off by combining a quality-seeking component, a diversity-seeking component and a selection component inspired from the Quality-Diversity literature. Crucially, we showed that quality and diversity could be optimized with off-policy reinforcement learning algorithms, resulting in a significantly better sample efficiency. We showed experimentally the effectiveness of the resulting QD-RL framework, which can solve, in two days using 10 CPUs, problems that were previously out of reach without a much larger infrastructure. Key components of QD-RL are selection through a Pareto front or a Map-Elites grid and the search for diversity in a behavior descriptor space. Admittedly, the behavior descriptor space needed to compute the diversity reward is hand designed. There are attempts to automatically obtain a relevant behavior space through unsupervised learning methods (Péré et al., 2018; Paolo et al., 2019), but defining this space is often a trivial decision that can alleviate the need to carefully design reward functions. In the future, we intend to extend this approach to problems where the environment reward function can be decomposed into several loosely dependent components, such as standing, moving forward and manipulating objects for a humanoid agent, or to multi-agent reinforcement learning problems. In such environments, we could replace the maximization of the sum of reward contributions with a multi-criteria selection from a Pareto front, where diversity would be only one of the considered criteria.

REFERENCES

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

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018.

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. In *Advances in neural information processing systems*, pp. 5027–5038, 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 and Jean-Baptiste Mouret. Behavioral repertoire learning in robotics. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 175–182, 2013.

Thang Doan, Bogdan Mazoure, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Attraction-repulsion actor-critic for continuous control reinforcement learning. *arXiv preprint arXiv:1909.07543*, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *Proc. of ICLR*, 2018.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Riashat Islam, Zafarali Ahmed, and Doina Precup. Marginalized state distribution entropy regularization in policy optimization. *arXiv preprint arXiv:1912.05128*, 2019.

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*, 2020.

Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. In *Neural Information Processing Systems*, 2018.

Shauharda Khadka, Somdeb Majumdar, Santiago Miret, Evren Tumer, Tarek Nassar, Zach Dwiel, Yinyin Liu, and Kagan Tumer. Collaborative evolutionary reinforcement learning. *arXiv preprint arXiv:1905.00976*, 2019.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011a.

Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 211–218, 2011b.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. PBCS: Efficient exploration and exploitation using a synergy between reinforcement learning and motion planning. *arXiv preprint arXiv:2004.11667*, 2020.

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

Giuseppe Paolo, Alban Laflaquiere, Alexandre Coninx, and Stephane Doncieux. Unsupervised learning and exploration of reachable outcome space. *algorithms*, 24:25, 2019.

Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Effective diversity in population-based reinforcement learning. *arXiv preprint arXiv:2002.00632*, 2020.

Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*, 2018.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Aloïs Pourchot and Olivier Sigaud. Cem-rl: Combining evolutionary and gradient-based methods for policy search. *arXiv preprint arXiv:1810.01222*, 2018.

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.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Longxiang Shi, Shijian Li, Qian Zheng, Min Yao, and Gang Pan. Efficient novelty search through deep reinforcement learning. *IEEE Access*, 8:128809–128818, 2020.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 30th International Conference in Machine Learning*, 2014.

Christopher Stanton and Jeff Clune. Curiosity search: producing generalists by encouraging individuals to continually explore and acquire skills throughout their lifetime. *PloS one*, 11(9):e0162235, 2016.

APPENDICES

A THE TD3 AGENT

The Twin Delayed Deep Deterministic (TD3) agent Fujimoto et al. (2018) builds upon the Deep Deterministic Policy Gradient (DDPG) agent Lillicrap et al. (2015). It trains a deterministic actor $\pi_\phi : \mathcal{S} \to \mathcal{A}$ directly mapping observations to continuous actions and a critic $Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ taking a state s and an action a and estimating the average return from selecting action a in state s and then following policy π_ϕ. As DDPG, TD3 alternates policy evaluation and policy improvement operations so as to maximise the average discounted return. In DDPG, the critic is updated to minimize a temporal difference error during the policy evaluation step which induces an overestimation bias. TD3 corrects for this bias by introducing two critics Q_{θ_1} and Q_{θ_2}. TD3 plays one step in the environment using its deterministic policy and then stores the observed transition (s_t, a_t, r_t, s_{t+1}) into a replay buffer $\mathcal{B}$. Then, it samples a batch of transitions from $\mathcal{B}$ and updates the critic networks. Half the time it also samples another batch of transitions to update the actor network.

Both critics are updated so as to minimize a loss function which is expressed as a mean squared error between their predictions and a target:

$$L^{critic}(\theta_1, \theta_2) = \sum_{\text{batch}} (Q_{\theta_1}(s_t, a_t) - y_t)^2 + (Q_{\theta_2}(s_t, a_t) - y_t)^2, \tag{2}$$

where the common target y_t is computed as:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\theta_i}(s_{t+1}, \pi_\phi(s_{t+1}) + \epsilon), \;\; \epsilon \sim \mathcal{N}(0, I). \tag{3}$$

The Q-value estimation used to compute target y_t is taken as minimum between both critic predictions thus reducing the overestimation bias. TD3 also adds a small perturbation ϵ to the action $\pi_\phi(s_{t+1})$ so as to smooth the value estimate by bootstrapping similar state-action value estimates.

Every two critics updates, the actor π_ϕ is updated using the deterministic policy gradient also used in DDPG Silver et al. (2014). For a state s, DDPG updates the actor so as to maximise the critic estimation for this state s and the action $a = \pi_\phi(s)$ selected by the actor. As there are two critics in TD3, the authors suggest to take the first critic as an arbitrary choice. Thus, the actor is updated by minimizing the following loss function:

$$L^{actor}(\phi) = -\sum_{\text{batch}} Q_{\theta_1}(s_t, \pi_\phi(s_t)). \tag{4}$$

Policy evaluation and policy improvement steps are repeated until convergence. TD3 demonstrates state of the art performance on several MuJoCo benchmarks. In this study, we use it to update the population of actors for both quality and diversity.

B QD-RL ADDITIONAL PRACTICAL DETAILS

B.1 COMPUTATIONAL DETAILS

We consider populations of $N = 4$ actors for both POINT-MAZE and POINT-MAZE-INERTIA environments and $N = 10$ actors for ANT-MAZE and ANT-TRAP. We use 1 CPU thread per actor and parallelization is implemented with the Message Passing Interface (MPI) library. Our experiments were run on a standard computation machine with 10 CPU cores and 100 GB of RAM although the maximum RAM consumption per experiment at any time never exceeds 10GB due to an efficient and centralized management of the archive that stores all solutions. An experiment on POINT-MAZE or POINT-MAZE-INERTIA typically takes between 2 and 3 hours while an experiment on ANT-MAZE or ANT-TRAP takes about 2 days. Note that these durations can vary significantly depending on the type of CPU used. We did not use any GPUs.

B.2 QD IMPLEMENTATION DETAILS

We consider two settings for the Quality Diversity part of our algorithm.

In the first setting, we rely on a FIFO archive of solutions. We chose a maximum archive size of 10.000 for all environments. When a new solution θ is obtained after the mutation phase, we compute the mean distance between its BD bd_θ and the BDs of its K nearest neighbors in the archive. We used $K = 10$ for all environments. We also used an Euclidean distance for all environments. We add the new solution in the archive if the computed distance is superior to a threshold of acceptance. See Table 3 for the different values of this threshold depending on the environment. During the selection phase, we compute a QD Pareto front of all solutions contained in the archive and sample the new population from the front. If the Pareto front contains less than N actors, we select them all, remove them, compute the Pareto front over the remaining actors and sample again from it, and so on until we get N actors.

In the second setting, we use a Map Elites (ME) grid as archive of solutions. We assume that the BD space is bounded and can be discretized in an Cartesian grid. We discretize each dimension into m meshes, see Table 3 for the value of m depending on the environment. Thus the number of cells in the ME grid equals m times the number of dimensions of the BDs. When a new solution θ is obtained after the mutation phase, we seek for the cell that corresponds its BD bd_θ. If the cell is empty, we add the solution in it, otherwise we replace the solution already contained by the new one if its score $\mathcal{J}(\theta)$ is greater than the one of the already contained solution. During the selection phase, we sample solutions uniformly from the ME grid. The ME archive management and selection mechanism tends to store less solutions than the method used in the first setting and does not need to perform heavy computations such as computing a Pareto Front which makes QD-RL ME less expensive in terms of memory and computing time than QD-RL PARETO.

B.3 DIVERSITY REWARD COMPUTATION

The RL part of QD-RL optimizes solutions for quality but also for diversity at the state level. As the QD part stores solutions and performs population selection based on solution BDs bd_θ, the RL part optimizes the solutions so as to encourage them to visit states with novel state BDs. The novelty of a state BD bd_t is expressed through a diversity reward r_t^D. In practice, we maintain in parallel of the replay buffer a FIFO storage of the states BDs encountered so far. In practice, the maximum size of this storage is never reached. However its maximum theoretical size is chosen as the same than the solutions archive. When a transition $(s_t, a_t, bd_t, r_t, s_{t+1})$ is sampled and stored in the Replay Buffer, we also add bd_t in this storage. As for the archive of solutions presented in the first QD setting, we add a state BD in this storage only if its mean Euclidean distance to its K nearest neighbors in the storage is greater than an acceptance threshold. The values of K and of the threshold are the same as the one used in the first QD setting. This mechanism ensures that we do not keep too close state BDs and thus controls this storage size. When a batch of transitions $(s_t, a_t, bd_t, r_t, s_{t+1})$ is collected during the mutation phase, we recompute fresh diversity rewards r_t^D as the mean Euclidean distance between the sampled states BDs bd_t and their K nearest neighbors in the states BDs storage. The value of K is the same as the one used for the insertion rule.

B.4 HYPER-PARAMETERS

Table 3 summarizes all hyper-parameters used in experiments (see Figure 4). Most of these hyper-parameters values are the original ones from the TD3 algorithm.

Table 3: QD-TD3 Hyper-parameters

Parameter	**Point Maze**	**Ant Maze**	**Ant Trap**
TD3			
Optimizer	Adam	Adam	Adam
Learning rate	6.10^{-3}	3.10^{-4}	3.10^{-4}
Discount factor γ	0.99	0.99	0.99
Replay buffer size	10^6	5.10^5	5.10^5
Hidden layers size	64/32	256/256	256/256
Activations	ReLU	ReLU	ReLU
Minibatch size	256	256	256
Target smoothing coefficient	0.005	0.005	0.005
Delay policy update	2	2	2
Target update interval	1	1	1
Gradient steps ratio	4	0.1	0.1
Quality-Diversity			
Archive size	10000	10000	10000
Threshold of acceptance	0.0001	0.1	0.1
K-nearest neighbors	10	10	10
MAP-Elites			
Number of cells per dimension	5	7	6

C ENVIRONMENTS ANALYSIS

In this section, we propose a more thorough analysis of the proposed environments. We highlight why these environments are hard to solve for classical deep RL agents without extra exploration mechanisms and show the impact of the different components of our algorithm.

C.1 ENVIRONMENTS DETAILS

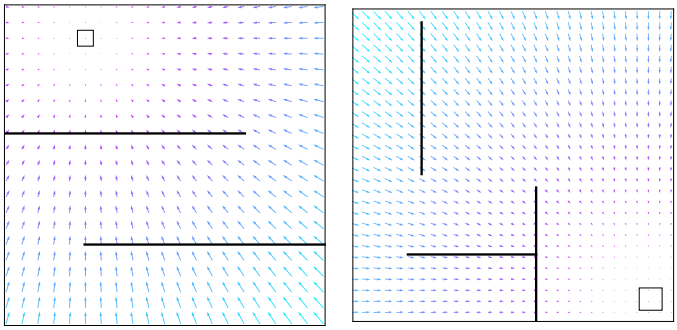
In the POINT-MAZE environment, the observation and state BD are the agent coordinates (x_t, y_t) at time t. The two continuous actions $(\delta x, \delta y)$ are position increments along the x and y axes. The solution BD is the final position (x_f, y_f) of the agent, as in Conti et al. (2018). The initial position of the agent is sampled from a small zone at the bottom of the maze. The exit area is a square located at the top left of the maze, the episode ends once this exit square is reached or after 200 time steps. The reward is computed as minus the Euclidean distance between the agent position and the maze exit. This reward leads to a deceptive gradient signal: following it would lead the agent to stay stuck by the second wall in the maze, as shown in Figure 6 of Appendix C. In POINT-MAZE-INERTIA, the BD space remains the same however the point velocity (v_x, v_y) is appended to the observation and two continuous actions correspond to forces (F_x, F_y).

In POINT-MAZE and POINT-MAZE-INERTIA, the dimensions of the observation spaces are equal to 2 and 4 respectively, and the dimensions of the action spaces are both equal to 2. We recall that these two environments are very similar with low-dimensional observation and action spaces, and only differ by the fact that POINT-MAZE-INERTIA moves a ball that is subject to inertia, thus the need to include the ball velocity in the observation space to preserve the Markov property. By contrast, in ANT-MAZE and ANT-TRAP, the dimensions of the observation spaces are respectively 29 and 113 while the action spaces dimensions are both equal to 8, making these two environments much more challenging to solve as they require larger controllers.

The ANT-TRAP environment also differs from mazes as it is open-ended, i.e., the space to be explored by the agent is unlimited, in opposition to other environments where this space is restricted by the maze walls. In this case, the state BD corresponds to the ant position that is clipped to remain in a given range. On the y-axis, this range is defined so as to leave a space corresponding to the width of the trap on both trap sides. On the x-axis, this range begins slightly behind the starting position of the ant and is large enough to let it accelerate along this axis. Figure 5 proposes a visual representation of the BD space in ANT-TRAP.

C.2 DECEPTIVE GRADIENTS

Figure 6 highlights the deceptive nature of the POINT-MAZE and the ANT-MAZE objective functions by depicting gradient fields in both environments. Similarly, POINT-MAZE-INERTIA and ANT-TRAP also present strongly deceptive gradients.



(a) Gradient field in POINT-MAZE (b) Gradient field in ANT-MAZE

Figure 6: Gradients maps on POINT-MAZE and ANT-MAZE. The black lines represent the maze walls, the arrows depict gradient fields and the square indicates the maze exit. Both settings present highly deceptive gradients: naively following them can only lead into a wall.

C.3 EXPLORATION IN POINT-MAZE FOR ALL ABLATIONS

Figure 7 summarizes the coverage maps of the POINT-MAZE environment by the different algorithms presented in the ablation study (see Table 1). A dot in the figure corresponds to the final position of an agent after an episode. The color spectrum highlights the course of training: agents evaluated early in training are in blue while newer ones are represented in purple. Figure 7 corresponds to the map coverage for one seed, we chose a very representative seed.

QD-TD3 (Pareto and MAP-Elites) and D-TD3+PARETO almost cover the totality of the BD space including the objective. D-TD3 also finds the objective but is not constant through all seeds and does not uniformly cover the BD space. Unsurprisingly, Q-TD3+PARETO and Q-TD3 present very poor map coverages, both algorithms optimize only for quality and the Pareto selection mechanism does not contribute anything in this setting. Interestingly, all algorithms optimizing for diversity find the maze exit. However, as shown in Table 1, algorithms that also optimize for quality (QD-TD3) or that present structural pressure toward quality (D-TD3+PARETO) are able to refine their trajectory through the maze and obtain significantly better performance.

Figure 8 summarizes the coverage maps of the ANT-TRAP environment by QD-TD3 MAP-ELITES, QD-TD3 PARETO and TD3. Interestingly, the pattern of QD-TD3 MAP-ELITES differs from that of QD-TD3 PARETO. We hypothesize that this behavior reflects the grid structure of the ME archive.

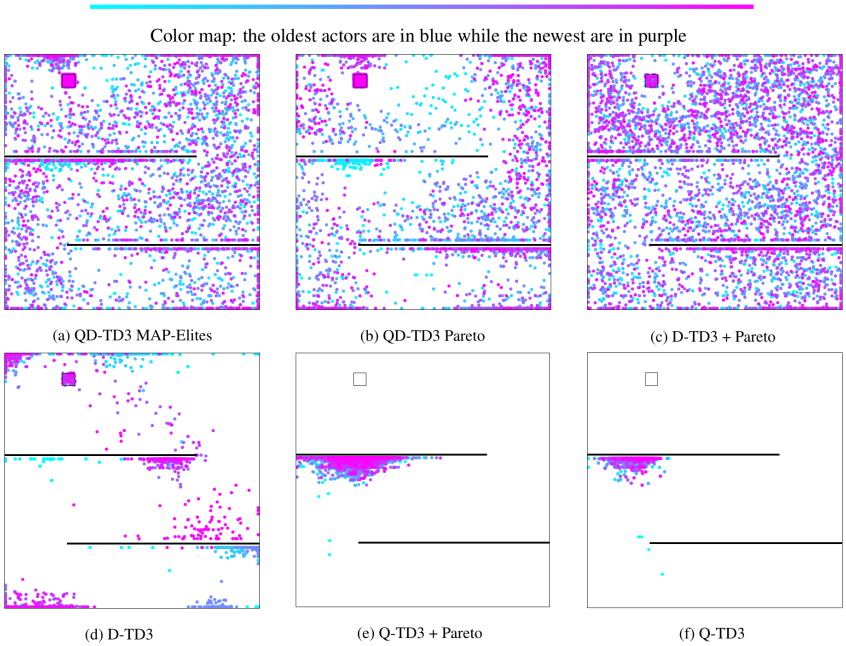


Figure 7: Coverage map of the POINT-MAZE environment for all ablations. Each dot corresponds to the position of an agent at the end of an episode. Dots corresponding to the oldest actors are in blue while the newest are in purple.

Figure 8: Coverage map of the ANT-TRAP environment. Each dot corresponds to the position of an agent at the end of an episode. Dots corresponding to the oldest agents are in blue while the newest are in purple.

D QD-RL PSEUDO CODE

Algorithm 1: QD-RL

Given: N, max_steps, gradient_steps_ratio,
Initialize: Archive A, Replay Buffer $\mathcal{B}$, N actors $\{\pi_{\theta_i}\}_{i=\{1,\dots,N\}}$, 2 critics Q_{θ^D} and Q_{θ^Q}, BD Storage S

$total_steps, actor_steps = 0, 0$ `// Step counters`

`// Parallel evaluation of the initial population`
for $j \leftarrow 1$ **to** N **do**
  Play one episode with actor π_{θ_j} and store all transitions in $\mathcal{B}$
  Get episode length T, discounted return R and state BDs $\{bd_1, \dots, bd_T\}$
  Store BDs $\{bd_1, \dots, bd_T\}$ in S
  Compute $bd_\theta = f_{bd}(\{bd_1, \dots, bd_T\})$ and add the tuple (R, bd_θ, θ_j) in the archive A
  $actor_steps \leftarrow actor_steps + T$
end

`// Algorithm main loop`
while $total_steps < max_steps$ **do**
  `// Select new generation`
  Get N actors $\pi_{\theta_i}, i \in \{1, \dots, N\}$ from the Pareto front or from the MAP-Elites grid
  $gradient_steps = int(actor_steps \times gradient_steps_ratio)$
  $actor_steps = 0$

  `// Perform in parallel population update and evaluation`
  for $j \leftarrow 1$ **to** N **do**
    `// Update the population`
    for $i \leftarrow 1$ **to** $gradient_steps$ **do**
      Sample batch of $(s_t, a_t, r_t, s_{t+1}, bd_t)$ from $\mathcal{B}$

      `// First half is updated to maximise diversity`
      if $j \leq N//2$ **then**
        Compute novelty reward as r_t^D from bd_t and S
        Update π_{θ_j} for diversity
        Compute novelty critic gradient locally
        Average between threads novelty critic gradients
        Update novelty critic Q_w^D
      end

      `// Second half is updated to maximise quality`
      else
        Update π_{θ_j} for quality
        Compute quality critic gradient locally
        Average between threads quality critic gradients
        Update quality critic Q_w^Q
      end
    end

    `// Evaluate the updated actors`
    Play one episode with actor π_{θ_j} and store all transitions in $\mathcal{B}$
    Get episode length T, discounted return R and state BDs $\{bd_1, \dots, bd_T\}$
    Store BDs $\{bd_1, \dots, bd_T\}$ in S
    Compute $bd_\theta = f_{bd}(\{bd_1, \dots, bd_T\})$ and add the tuple (R, bd_θ, θ_j) in the archive A
    $actor_steps \leftarrow actor_steps + T$
  end

  $total_steps \leftarrow total_steps + actor_steps$ `// Update total time steps`

end