DIVERSITY POLICY GRADIENT FOR SAMPLE EFFICIENT QUALITY-DIVERSITY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A fascinating aspect of nature lies in its ability to produce a large and diverse collection of high-performing organisms in an open-ended way. By contrast, most AI algorithms seek convergence and focus on finding a single efficient solution to a given problem. Aiming for diversity through divergent search in addition to performance is a convenient way to deal with the exploration-exploitation trade-off that plays a central role in learning. It also allows for increased robustness when the returned collection contains several working solutions to the considered problem, making it well-suited for real applications such as robotics. Quality-Diversity (QD) methods are evolutionary algorithms designed for this purpose. This paper proposes a novel algorithm, QD-PG, which combines the strength of Policy Gradient algorithms and Quality Diversity approaches to produce a collection of diverse and high-performing neural policies in continuous control environments. The main contribution of this work is the introduction of a Diversity Policy Gradient (DPG) that drives policies towards more diversity in a sample-efficient and open-ended manner. Specifically, QD-PG selects neural controllers from a MAP-ELITES grid and uses two gradient-based mutation operators to improve both quality and diversity. Our results demonstrate that QD-PG is significantly more sample-efficient than its evolutionary competitors.

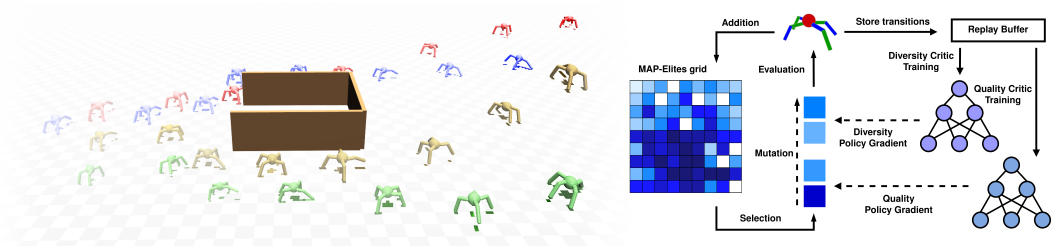


Figure 1: An agent robot is rewarded for running forward as fast as possible, following the reward signal without further exploration leads the agent into a trap. Our method QD-PG allows deeper exploration, necessary to solve such deceptive control problems. QD-PG builds upon the MAP-ELITES and leverages the data-efficiency of reinforcement learning by using a quality critic and a diversity critic to derive policy gradient based mutations in order to seek for high performance and new behaviors.

1 INTRODUCTION

Natural evolution has the fascinating ability to produce diverse organisms that are all well adapted to their respective niche. It is a creative and open-ended (OE) process that continuously produces new solutions without explicitely optimizing for a fitness objective. Such open-endedness could be the key to solving problems where the required stepping stones cannot be anticipated and planned, emphasizing the need for a serendipitous search process. Inspired by this ability to produce a tremendous diversity of living systems, Quality-Diversity (QD) algorithms aim at searching for a collection of both diverse and high-performing solutions (Pugh et al., 2016; Cully & Demiris, 2017). While

classic optimization algorithms focus on finding a single efficient solution, QD algorithms aim to cover the range of possible solution types and to return the best solution for each type.

By seeking diversity, QD algorithms can be viewed as partially open-ended tools that balance open-endedness with practical use (Mouret, 2020). Diversity search is the core component that allows QD algorithms to generate large collections of diverse solutions in an open-ended manner. By continuously encouraging the emergence of novel solutions never seen before, the search for diversity is only limited by the very nature of the considered problem (or environment). QD methods have been useful in autonomous skill discovery (Cully, 2019), demonstrating their ability to create solutions of increasing complexity from the void, but also as a part of broader open-ended systems (Wang et al., 2019).

Building large and efficient controllers that work with continuous actions has been a long-standing goal in Artificial Intelligence and particularly in robotics. Deep reinforcement learning (RL), and especially Policy Gradient (PG) methods have proven efficient at training such large controllers (Schulman et al., 2017; Lillicrap et al., 2015; Fujimoto et al., 2018; Haarnoja et al., 2018). One of the keys to this success lies in the fact that PG methods exploit the structure of the objective function when the problem can be formalized as a Markov Decision Process (MDP). Moreover, they also exploit the analytical structure of the controller when known, which allows the sample complexity of these methods to be independent of the parameter space dimensionality (Vemula et al., 2019).

Although exploration is very important to reach optimal policies, PG methods usually rely on simple exploration mechanisms, like adding Gaussian noise (Fujimoto et al., 2018) or maximizing entropy (Haarnoja et al., 2018) to explore the action space, which happens to be insufficient in hard exploration tasks where the reward signal is sparse or deceptive (Colas et al., 2018; Nasiriany et al., 2019). These techniques only focus on building high-performing solutions and do not explicitly encourage diversity within the set of produced solutions. In this regard, they fail when confronted with hard exploration problems.

Contributions. In this work, we introduce the idea of a *diversity policy gradient* (D-PG) that drives the search process towards unseen and novel solutions in an open-ended way. We show that the D-PG can be used in combination with the standard policy gradient, dubbed *quality policy gradient* (Q-PG), to produce high-performing and diverse solutions. Our algorithm, called QD-PG, builds on MAP-ELITES and PGA-ME (Nilsson & Cully, 2021) replacing random diversity search by D-PG, and demonstrates remarkable sample efficiency brought by off-policy PG methods. We compare QD-PG to state-of-the-art policy gradient methods algorithms (including SAC, TD3, RND, AGAC, DIAYN and PGA-ME), and to several evolutionary methods known as Evolution Strategies (ESs) augmented with a diversity objective (namely the NS-ES family (Conti et al., 2018) and the ME-ES algorithm (Colas et al., 2020)) on a set of challenging continuous control tasks with deceptive reward signals.

2 BACKGROUND

2.1 PROBLEM STATEMENT

We consider an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the dynamics transition function and γ a discount factor. We assume that both $\mathcal{S}$ and $\mathcal{A}$ are continuous and consider a controller, or policy, $\pi_\theta : \mathcal{S} \to \mathcal{A}$, a neural network parameterized by $\theta \in \Theta$, which is called a *solution* to the problem. The *fitness* $F : \Theta \to \mathbb{R}$ of a solution measures its performance, defined as the expectation over the sum of rewards obtained by controller π_θ. A solution with high fitness is said to be *high-performing*.

We introduce a behavior descriptor (BD) space $\mathcal{B}$, a behavior descriptor extraction function $\xi : \Theta \to \mathcal{B}$, and define a distance metric $||.||_\mathcal{B}$ over $\mathcal{B}$. The *diversity* of a set of K solutions $\{\theta_k\}_{k=1,\dots,K}$ is defined as $d : \Theta^K \to \mathbb{R}^+$:

$$d\left(\{\theta_k\}_{k=1,\dots,K}\right) = \sum_{i=1}^{K} \min_{k \neq i} ||\xi(\theta_i), \xi(\theta_k)||_\mathcal{B}, \tag{1}$$

meaning that a set of solutions is diverse if the solutions are distant with respect to each other in the sense of $||.||_\mathcal{B}$.

Following the objective of the family of Quality Diversity algorithms, we are trying to evolve a population of *diverse* and *high-performing* solutions.

2.2 THE MAP-ELITES ALGORITHM

MAP-ELITES (Mouret & Clune, 2015) is a simple yet state-of-the-art QD algorithm that has been successfully applied to a wide range of challenging problems such as robot damage recovery (Cully et al., 2015), molecular robotic control (Cazenille et al., 2019) and game design (Alvarez et al., 2019). In MAP-ELITES, the behavior descriptor space $\mathcal{B}$ is discretized into a grid of cells, also called niches, with the aim of filling each cell with a high-performing solution. A variant, called CVT MAP-ELITES uses Centroidal Voronoi Tesselations (Vassiliades et al., 2016) to initially divide the grid into the desired number of cells. The algorithm starts with an empty grid and an initial random set of K solutions that are evaluated and added to the grid by following simple insertion rules. If the cell corresponding to the behavior descriptors of a solution is empty, then the solution is added to this cell. If there is already a solution in the cell, the new solution replaces it only if it has greater fitness. At each iteration, P existing solutions are sampled uniformly from the grid and randomly mutated to create P new solutions, encouraging the emergence of serendipity throughout the search process: a new high-performing solution is often created from a distant stepping stone in the behavior space. These new solutions are then evaluated and added to the grid following the same insertion rules. This cycle is repeated until convergence or for a given budget of iterations.

2.3 REINFORCEMENT LEARNING AND TD3

Deep Reinforcement learning (DRL) is a paradigm to learn high-performing policies implemented by neural networks in MDPs. In this work, we focus on a class of policy search methods called policy gradient methods. In opposition to standard evolutionary methods that rely on random updates, policy gradient methods exploit the structure of the MDP under the form of the Bellman equations to compute efficient performance-driven updates to improve the policy.

Among many algorithms in the reinforcement learning community, TD3 (Fujimoto et al., 2018) shows state-of-the-art performance to train controllers in environments with continuous action space and large state space. TD3 relies on the deterministic policy gradient update Silver et al. (2014) to train deterministic policies, as defined in Section 2.1. In most policy gradients methods, a critic $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ implemented by a neural network is introduced. The critic evaluates the expected fitness of policy π when performing a rollout starting from a state-action pair (s, a). The policy is updated to take the actions that will maximise the critic's value estimation in each state. Both the actor and the critic have an associated target network and use delayed policy updates to stabilize the training. More details can be found in the supplementary material in Section B.

3 KEY PRINCIPLE: DIVERSITY POLICY GRADIENT

3.1 GENERAL PRINCIPLE

In this work, we introduce the Diversity Policy Gradient, which aims at using time-step level information in order to mutate controllers towards diversity. The DPG acts in addition to the structural pressure for diversity induced by the MAP-ELITES selection and insertion mechanisms. It can be viewed as a divergent search (Lehman & Miikkulainen, 2015), open-ended oriented component, which continuously drives the search process towards creating solutions that are novel and different from the so far created ones. We are trying to characterize the behavior of policies thanks to the states they visit and hence construct a diversity reward at the time-step-level that is correlated with the diversity at the episode-level. This reward is computed based on distance of a state compared to the nearest states visited by other controllers in the MAP-ELITES grid. Aiming at maximizing the accumulated diversity rewards will induce an increase of the diversity of the population we are evolving. The following section provides a mathematical motivation for this approach.

3.2 MATHEMATICAL FORMULATION

In this section, we motivate and introduce formally the D-PG computations.

Let us assume that we have a MAP-ELITES grid containing K solutions $(\theta_1, \dots, \theta_K)$ and that we sampled θ_1 from the grid to evolve it. We want to update θ_1 in such a way that the population's diversity as defined in Equation equation 1 will increase. For this purpose, we aim to compute the gradient of the diversity with respect to θ_1 and update θ_1 in its direction using standard gradient ascent techniques.

Proposition 1 *The gradient of diversity with respect to θ_1 can be written as*

$$\begin{cases} \nabla_{\theta_1} d(\{\theta_k\}_{k=1,\dots,K}) = \nabla_{\theta_1} n(\theta_1, (\theta_j)_{2\leq j\leq J}), \\ \textit{where } \; n(\theta_1, (\theta_j)_{2\leq j\leq J}) = \sum_{j=2}^{J} ||\xi(\theta_1), \xi(\theta_j)||_{\mathcal{B}} \end{cases} \tag{2}$$

and θ_2 is θ_1 closest neighbour and (θ_j) with $j = 3, \dots, J$ are the elements in the population for which θ_1 is the nearest neighbour. Proof in Appendix F.

We call n the novelty of θ_1 with respect to its nearest elements. This proposition means that we can increase the diversity of the population by increasing the novelty of θ_1 with respect to the solutions for which it is the nearest neighbor.

Under this form, the diversity gradient cannot benefit from the variance reduction methods in the RL literature to efficiently compute policy gradients Sutton et al. (1999). To this end, we need to express it as a gradient over the expectation of a sum of scalar quantities obtained by policy π_{θ_1} at each step when interacting with the environment.

For this purpose, we introduce a novel space $\mathcal{D}$, dubbed *state descriptor space* and a *state descriptor extraction function* $\psi : \mathcal{S} \to \mathcal{D}$. We assume $\mathcal{D}$ and $\mathcal{B}$ have the same dimension. The notion of state descriptor will be used in the following to link diversity at the time step level to the diversity at the trajectory level.

In this context, if the following compatibility equation is satisfied:

$$\begin{cases} n(\theta_1, (\theta_j)_{2\leq j\leq J}) = \mathbb{E}_{\pi_{\theta_1}} \sum_t n(s_t, (\theta_j)_{2\leq j\leq J}) \\ \text{where } n(s, (\theta_j)_{j=1,\dots,J}) = \sum_{j=1}^{J} \mathbb{E}_{\pi_{\theta_j}} \sum_t ||\psi(s), \psi(s_t)||_{\mathcal{D}} \end{cases} \tag{3}$$

then the diversity policy gradient can be computed as:

$$\nabla_{\theta_1} d(\{\theta_k\}_{k=1,\dots,K}) = \nabla_{\theta_1} \mathbb{E}_{\pi_{\theta_1}} \sum_t n(s_t, (\theta_j)_{2\leq j\leq J}) \tag{4}$$

See the Appendix F for more details about the motivation behind this assumption. The obtained expression corresponds to the classical policy gradient setting where $\gamma = 1$ and where the corresponding reward signal, here dubbed diversity reward, is computed as $r_t^D = n(s_t, (\theta_j)_{\ 2\leq j\leq J})$. Therefore, this gradient can be computed using any PG estimation technique replacing the environment reward by the diversity reward r_t^D.

Equation equation 3 enforces a relation between $\mathcal{B}$ and $\mathcal{D}$ and between extraction functions ψ and ξ. In practice, it may be hard to define the behavior descriptor and state descriptor of a solution that satisfy this relation while being meaningful to the problem at hand and tractable. Nevertheless, a strict equality is not necessary: a positive correlation between the two hand-size is sufficient for diversity seeking.

4 Related Work

Simultaneously maximizing diversity and performance is the central goal of QD methods (Pugh et al., 2016; Cully & Demiris, 2017). Among the various possible combinations offered by the QD framework (Cully & Demiris, 2017), Novelty Search with Local Competition (NSLC) (Lehman & Stanley, 2011b) and MAP-ELITES (Mouret & Clune, 2015) are the two most popular algorithms.

NSLC builds on the Novelty Search (NS) algorithm (Lehman & Stanley, 2011a) and maintains an unstructured archive of solutions selected for their local performance while MAP-ELITES uniformly samples individuals from a structured grid that discretizes the BD space. Although very efficient in small parameter spaces, those methods struggle to scale to bigger spaces which limits their application to neuro-evolution.

Gradients in QD. Algorithms such as NSR-ES and NSRA-ES have been applied to challenging continuous control environments (Conti et al., 2018). But, as outlined by Colas et al. (2020), they still suffer from poor sample efficiency and the diversity and environment reward functions could be mixed in a more efficient way. ME-ES (Colas et al., 2020) went one step further in that direction, achieving a better mix of quality and diversity seeking through the use of a MAP-ELITES grid and two specialized ES populations. Using these methods was shown to be critically more successful than population-based GA algorithms (Salimans et al., 2017), but they rely on heavy computation resources. Differentiable QD (Fontaine & Nikolaidis, 2021) improves data-efficiency in QD with an efficient search in the descriptor space but does not tackle neuro-evolution and is limited to problems where the fitness and behavior descriptor functions are differentiable.
Exploration and diversity in RL. RL methods generally seek for diversity either in the state space or in the action space. This is the case of algorithms maintaining a population of RL agents for exploration without an explicit diversity criterion (Jaderberg et al., 2017) or algorithms explicitly looking for diversity but in the action space rather than in the state space like ARAC (Doan et al., 2019), AGAC (Flet-Berliac et al.), P3S-TD3 (Jung et al., 2020) and DvD (Parker-Holder et al., 2020).

An exception is *Curiosity Search* Stanton & Clune (2016) which defines a notion of *intra-life novelty* that is similar to our state novelty defined in Section 3. However, their novelty relies on skills rather than states. Our work is also related to algorithms using RL mechanisms to search for diversity only like DIAYN (Eysenbach et al., 2018) and others (Pong et al., 2019; Lee et al., 2019; Islam et al., 2019). These methods have proven useful in sparse reward situations, but they are inherently limited when the reward signal can orient exploration, as they ignore it. Other works sequentially combine diversity seeking and RL (Colas et al., 2018; Ecoffet et al., 2019).

Mixing policy gradient and evolution. The fruitful synergy between evolutive and RL methods has been explored in many recent methods, notably ERL (Khadka & Tumer, 2018), CERL (Khadka et al.), CEM-RL (Pourchot & Sigaud, 2018) and PGA-ME (Nilsson & Cully, 2021). ERL and CEM-RL mix Evolution Strategies and RL to evolve a population of agents to maximize quality but ignores the diversity of the population. Policy Gradient Assisted MAP-Elites (PGA-ME) successfully combines QD and RL. This algorithm scales MAP-ELITES to neuroevolution by evolving half of its offsprings with a quality policy gradient update instead of using a genetic mutation alone. Nevertheless, the seek for diversity is only explicitly done with the genetic mutation.

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

5 METHODS

Our full algorithm is called QD-PG, its pseudo code and general architecture are given in Apendix A. QD-PG is an iterative algorithm based on MAP-ELITES that replaces random mutations with policy gradient updates. As we consider a continuous action space and want to improve sample efficiency by using an off-policy policy gradient method, we rely on TD3.

QD-PG maintains three permanent structures. On the QD side, a CVT MAP-ELITES grid stores the most novel and performing solutions. On the RL side, a replay buffer contains all transitions collected when evaluating solutions and an archive $\mathbb{A}$ stores all state descriptors obtained so far. QD-PG starts with an initial population of random solutions, evaluates them and inserts them into the MAP-ELITES grid. At each iteration, solutions are sampled from the grid, copied, and updated. The updated solutions are then evaluated through one rollout in the environment and inserted into the grid according to the usual insertion rules. Transitions collected during evaluation are stored in the replay buffer, and state descriptors are stored in the archive $\mathbb{A}$. Note that these state descriptors are

first filtered to avoid insertion in the archive of multiple state descriptors that are too close to each other.

During the update step, half the population is updated with Q-PG ascent and the other half with D-PG ascent. The choice of whether an agent is updated for quality or diversity is random, meaning that it can be updated for quality and later for diversity if selected again. To justify this design, we show in Section 7 that updating consecutively for quality and diversity outperforms updating based on joint criteria. Both gradients are computed from batches of transitions sampled from the replay buffer. The Q-PG is computed from the usual environment rewards (similar to TD3) whereas for D-PG, we get "fresh" novelty rewards as the sum of the distances between state s_t descriptor and its J nearest neighbors in the archive $\mathbb{A}$. Diversity rewards must be recomputed at each update because $\mathbb{A}$ changes during training. Following Equation equation 3, diversity rewards should be computed as the sum of the distances between the descriptor of s_t and the descriptors of all the states visited by a list of J solutions. In practice, we consider the J nearest neighbors of s_t. This choice simplifies the algorithm, is faster to compute and works well in practice.

TD3 relies on a parameterized critic to reduce the variance of its policy gradient estimate. In QD-PG, we maintain two parameterized critics Q_w^D and Q_v^Q, respectively dubbed diversity and quality critics. Every time a policy gradient is computed, QD-PG also updates the corresponding critic. The critics are hence trained with all agents in the population instead of a specialised agent. This helps avoiding local minima in exploration environments where the specialised actor could get stuck and hence mislead the values learned by the critic. However, having critics trained with multiple agents can destabilize the process, which is why we avoid using QD-PG with big grids. In our benchmarks, our grids usually contain 3 times less cells than PGA-ME. As in TD3, we use pairs of critics and target critics to fight the overestimation bias. We share the critic parameters among the population as in CEM Pourchot & Sigaud (2018). Reasons for doing so come from the fact that diversity is not stationary, as it depends on the current population. If each agent had its own diversity critic, since an agent may not be selected for a large number of generations before being selected again, its critic would convey an outdated picture of the evolving diversity. We tried this solution, and it failed. A side benefit of critic sharing is that both critics become accurate faster as they combine experience from all agents. Additional details on QD-PG implementation are available in Appendix C.

6 EXPERIMENTS

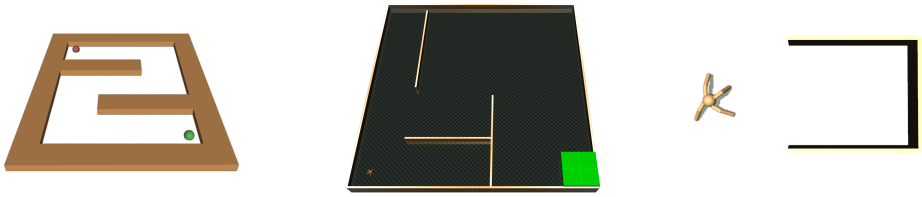
In this section, we intend to answer the following questions: 1. Can QD-PG produce collections of diverse and high-performing neural policies and what are the advantages to do so? 2. Is QD-PG more sample efficient than its evolutionary competitors? 3. How difficult are the considered benchmarks for classical policy gradients methods?

6.1 ENVIRONMENTS

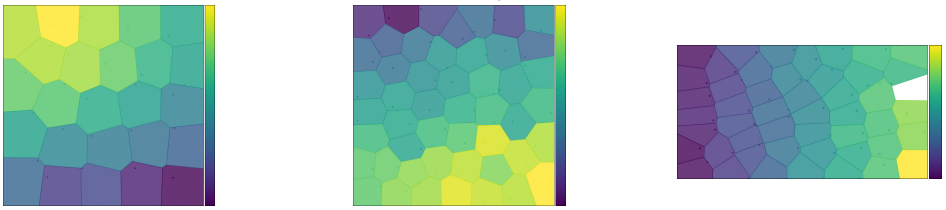
We assess QD-PG capabilities in continuous control environments that exhibit high dimensional observation and action spaces as well as deceptive rewards. The size of the state/action space makes exploration difficult for Genetic Algorithms and Evolution Strategies. Deceptive rewards creates exploration difficulties which is particularly challenging for classical RL methods. We consider three OpenAI Gym environments based on the MuJoCo physics engine that all exhibit strong deceptive rewards (illustrated in the Appendix in Figure 5), POINT-MAZE, ANT-MAZE and ANT-TRAP. Such environments have also been widely used in previous works (Parker-Holder et al., 2020; Colas et al., 2020; Frans et al., 2018; Shi et al., 2020).

In the POINT-MAZE environment, an agent represented as a green sphere must find the exit of the maze depicted in Figure 2a, represented as a red sphere. The reward is expressed as the negative Euclidean distance between the center of gravity of the agent and the exit center. The ANT-MAZE environment is modified from OpenAI Gym ANT-V2 (Brockman et al., 2016) and also used in (Colas et al., 2020; Frans et al., 2018). In ANT-MAZE, a four-legged ant starts in the bottom left of a maze and has to reach a goal zone located in the lower right part of it (green area in Figure 2a). As in POINT-MAZE, the reward is expressed as the negative distance the goal zone.

Finally, the ANT-TRAP environment also derives from ANT-V2 and is inspired from (Colas et al., 2020; Parker-Holder et al., 2020). In ANT-TRAP, the four-legged ant initially appears in front of a



(a) We tested QD-PG on 3 environments, from left to right: POINT-MAZE, ANT-MAZE and ANT-TRAP



(b) QD-PG final grid on POINT-MAZE, ANT-MAZE and ANT-TRAP respectively. Colors on final grids show fitness of solutions, yellow corresponding to high fitness and dark purple to low fitness.

Figure 2: Visual representation of POINT-MAZE, ANT-MAZE and ANT-TRAP and their respective final grids.

trap and must bypass it to run as fast as possible in the forward direction (see Figure 2a), as in ANT-V2, the reward is computed as the ant velocity on the x-axis. Additional details on environments can be found in Appendix D

7 RESULTS

Can QD-PG produce collections of neural policies and what are the advantages to do so? Table 1a presents QD-PG performances. In all environments, our algorithm manages to find working solutions that avoid local minima and reach the overall objective. In addition to its exploration capabilities, QD-PG generates collections of high performing solutions in a single run. During the ANT-TRAP experiment, the final collection of solutions returned by QD-PG contained, among others, 5 solutions that were within a 10% performance margin from the best one.

Generating a collection of diverse solutions comes with the benefit of having a repertoire of diverse solutions that can be used as alternatives when the MDP changes (Cully et al., 2015). We show that QD-PG is more robust than conventional policy gradient methods by changing the reward signal of the ANT-MAZE environment. We replace the original goal in the bottom right part of the maze (see Figure 3) with a new randomly located goal in the maze. Instead of running QD-PG to optimize for this new objective, we run a Bayesian optimization process to quickly find a good solution among the ones already stored in the grid. With a budget of only 20 solutions to be tested during the Bayesian optimization process, we are able to quickly recover a good solution for the new objective. We repeat this experiment 100 times, each time with a different random goal, and obtain an average performance of -10 with a standard deviation of 9. In other words, 20 interaction episodes (corresponding to 60.000 time steps) suffice for the adaptation process to find a solution that performs well for the new objective without the need to re-train agents. More detailed results can be found in Appendix E.4.

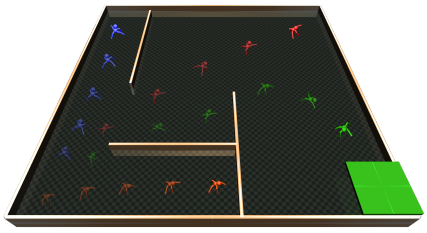


Figure 3: QD-PG produces a collection of diverse solutions. In ANT-MAZE, even after setting new randomly located goals, the MAP-ELITES grid still contains solutions that are suited for the new objectives.

Is QD-PG more sample efficient than its evolutionary competitors? Table 1b compares QD-PG to Deep Neuroevolution algorithms with a diversity seeking component (ME-ES, NSR-ES, NSRA-

Table 1: Results for all environments. **Final Perf.** is the minimum distance to the goal in ANT-MAZE and the episode return in POINT-MAZE and ANT-TRAP. The **Ratio to ours** column compares the sample efficiency of a method to QD-PG.

(a) Comparison to ablations and PG baselines.

	Final Perf. ($\pm$ std)		
Algorithm	POINT-MAZE	ANT-MAZE	ANT-TRAP
QD-PG	$\mathbf{-24(\pm 0)}$	$-7(\pm 7)$	$\mathbf{1541(\pm 86)}$
QD-PG SUM	$-24(\pm 0)$	$-5(\pm 3)$	$1018(\pm 6)$
D-PG	$-36(\pm 2)$	$\mathbf{-2(\pm 0)}$	$1016(\pm 8)$
Q-PG	$-128(\pm 0)$	$-26(\pm 0)$	$1175(\pm 79)$
SAC	$-127(\pm 1)$	$-59(\pm 1)$	$1049(\pm 21)$
TD3	$-130(\pm 2)$	$-26(\pm 0)$	$1131(\pm 7)$
RND	$-35(\pm 10)$	$-27(\pm 1)$	$978(\pm 61)$
CEM-RL	$-312(\pm 1)$	$-26(\pm 0)$	$934(\pm 22)$
P3S-TD3	$-144(\pm 14)$	$-60(\pm 0)$	$1173(\pm 4)$
AGAC	$-32(\pm 49)$	$-43(\pm 3)$	$1113(\pm 8)$
DIAYN	$-96(\pm 14)$	$-47(\pm 4)$	$949(\pm 34)$
PGA-ME	$-126(\pm 0)$	$-18(\pm 6)$	$1455(\pm 17)$

(b) Comparison to evolutionary competitors.

	ANT-MAZE		
Algorithm	**Final Perf.**	**Steps to goal**	**Ratio to ours**
QD-PG	$-1(\pm 7)$	$\mathbf{8.4e7}$	**1**
CEM-RL	$-26(\pm 0)$	∞	∞
ME-ES	$-5(\pm 1)$	$2.4e10$	286
NSR-ES	$-26(\pm 0)$	∞	∞
NSRA-ES	$\mathbf{-2(\pm 1)}$	$2.1e10$	249

ES, CEM-RL) in terms of sample efficiency. QD-PG runs on 10 CPU cores for 2 days while its competitors used 1000 CPU cores for the same duration.

We see three reasons for the improved sample efficiency of QD-PG: 1) QD-PG leverages a replay buffer and can re-use each sample several times. 2) QD-PG leverages novelty at the state level and can exploit all collected transitions to maximize quality and diversity. For instance, in ANT-MAZE, a trajectory brings 3000 samples to QD-PG while standard QD methods would consider it a unique sample. 3) PG exploits the analytical gradient between the neural network weights and the resulting policy action distribution and estimates only the impact of the distribution on the return. By contrast, standard QD methods directly estimate the impact on the return of randomly modifying the weights.

How challenging are the considered benchmarks for policy gradients methods? Table 1a compares QD-PG to state-of-the-art policy gradient algorithms and validates that classical policy gradient (TD3 and SAC) methods fail to find optimal solutions in deceptive environments. Besides, despite its exploration mechanism based on CEM, CEM-RL also quickly converges to local optima in all benchmarks, confirming the need for a dedicated diversity seeking component. RND, which adds an exploration bonus used as an intrinsic reward, also demonstrates performances inferior to QD-PG in all environments but manages to solve POINT-MAZE. In ANT-TRAP, RND extensively explores the BD space but fails to obtain high returns. Although maintaining diversity into a population of reinforcement learning agents, AGAC and P3S-TD3 are not able to explore enough the environments to solve them, showing some limits of exploration through the action space.

DIAYN is able to explore in POINT-MAZE but does not fully explore the environment and hence does not reach the goal before the end of the 10^6 steps. DIAYN ensures that its skills explore states different enough to be discriminated, but once they are, there is no incentive to further explore. Moreover, DIAYN in ANT-TRAP shows a limit of unsupervised learning methods: ANT-TRAP is an example of environment where the behavior descriptor chosen is not perfectly aligned with the performance measured, as a matter of fact, the objective is not to reach a point but to achieve the best forward speed with some control costs. But as those rewards are completely ignored by DIAYN, the produced controllers have very low fitness.

8 CONCLUSION

We introduced a diversity gradient to efficiently replace divergent search for diversity seeking. Based on this component, we proposed a novel algorithm, QD-PG, that builds upon MAP-Elites algorithm and significantly improve its sample efficiency. We believe that in future work, the DPG could be part of and enhance open-ended algorithms that seek diversity in a more sample efficient manner.

REFERENCES

Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2019.

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

Leo Cazenille, Nicolas Bredeche, and Nathanael Aubert-Kato. Exploring self-assembling behaviors in a swarm of bio-micro-robots using surrogate-assisted map-elites. *arXiv preprint arXiv:1910.00230*, 2019.

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. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 81–89, 2019.

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.

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.

Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, and Philippe Preux. Adversarially guided actor-critic.

Matthew C. Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. *CoRR*, abs/2106.03894, 2021. URL `https://arxiv.org/abs/2106.03894`.

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.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 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, Tarek Nassar, Zach Dwiel, Evren Tumer, Santiago Miret, Yinyin Liu, and Kagan Tumer. Collaborative evolutionary reinforcement learning.

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 Risto Miikkulainen. Enhancing divergent search through extinction events. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 951–958, 2015.

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.

Jean-Baptiste Mouret. Evolving the behavior of machines: from micro to macroevolution. *Iscience*, 23(11):101731, 2020.

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

Soroush Nasiriany, Vitchyr H Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. *arXiv preprint arXiv:1911.08453*, 2019.

Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites; policy gradient assisted map-elites. 2021. doi: 10.1145/3449639.3459304. URL `https://hal.archives-ouvertes.fr/hal-03135723v2`.

Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Effective diversity in population-based reinforcement learning. In *Neural Information Processing Systems*, 2020.

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.

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

Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pp. 1057–1063. Citeseer, 1999.

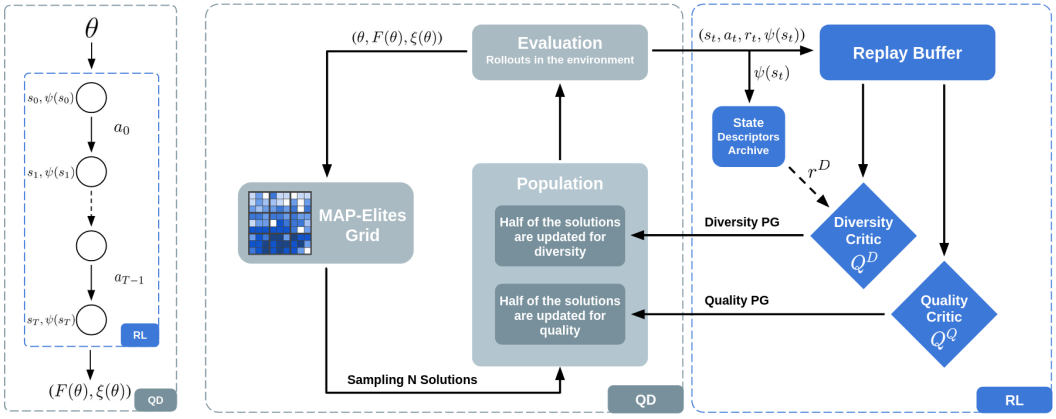
Vassilis Vassiliades, Konstantinos I. Chatzilygeroudis, and Jean-Baptiste Mouret. Scaling up map-elites using centroidal voronoi tessellations. *CoRR*, abs/1610.05729, 2016. URL `http://arxiv.org/abs/1610.05729`.

Anirudh Vemula, Wen Sun, and J Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2926–2935. PMLR, 2019.

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.

A QD-PG CODE AND ARCHITECTURE

The full algorithm can be found on pseudocode 1 and its architecture in Figure 4.



(a) (b)

Figure 4: (a): The RL part of QD-PG operates at the time step level while the QD part operates at the controller level, considering the MDP as a black box. (b) One QD-PG iteration consists of three phases: 1) A new population of solutions is sampled from the MAP-ELITES grid. 2) These solutions are updated by an off-policy RL agent: half of the solutions are optimized for quality and the other half for diversity. The RL agent leverages one shared critic for each objective. 3) The newly obtained solutions are evaluated in the environment. Transitions are stored in a replay buffer while the updated solutions, their final scores and behavior descriptors are stored in the MAP-ELITES grid.

B DETAILED EQUATIONS OF TD3

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 $\pi_\phi : \mathcal{S} \rightarrow \mathcal{A}$ directly mapping observations to continuous actions and a critic $Q_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ which takes a state s and an action a and estimates the average return from selecting action a in state s and then following policy π_ϕ. As DDPG, TD3 alternates between policy evaluation and policy improvement 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 $\mathcal{M}$. Then, it samples a batch of transitions from $\mathcal{M}$ 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:

$$L^{critic}(\theta_1, \theta_2) = \sum_{\text{batch}} \sum_{i=1,2} (Q_{\theta_i}(s_t, a_t) - y_t)^2, \tag{5}$$

where the common target y_t is computed as:

$$y_t = r_t + \gamma \min_{i=1,2} Q_{\theta_i}(s_{t+1}, \pi_\phi(s_{t+1}) + \epsilon), \tag{6}$$

where $\epsilon \sim \mathcal{N}(0, I)$.

Algorithm 1: QD-PG

Given: N, max_steps, gradient_steps_ratio, BD extraction function ξ, state descriptor extraction function ψ
Initialize: MAP-Elites grid $\mathbb{M}$, Replay Buffer $\mathbb{R}$, N actors $\{\pi_{\theta_i}\}_{i=\{1,\dots,N\}}$, 2 critics Q_w^D and Q_w^Q, state descriptors archive $\mathbb{A}$

$total_steps, actor_steps = 0, 0$ `// 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 $\mathbb{R}$
  Get episode length T, discounted return R and state descriptors $\{\psi(s_1), \dots, \psi(s_T)\}$
  Store state descriptors $\{\psi(s_1), \dots, \psi(s_T)\}$ in $\mathbb{A}$
  Compute $\xi(\theta_j)$ and add the tuple $(R, \xi(\theta_j), \theta_j)$ in the MAP-Elites grid $\mathbb{M}$
  $actor_steps \leftarrow actor_steps + T$
end

`// Main loop`
while $total_steps < max_steps$ **do**
  `// Select new generation`
  Get N actors $\pi_{\theta_i}, i \in \{1, \dots, N\}$ from $\mathbb{M}$
  $gradient_steps = int(actor_steps \times gradient_steps_ratio)$
  $actor_steps = 0$

  `// 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 $(s_t, a_t, r_t, s_{t+1}, \psi(s_t))$ from $\mathbb{R}$

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

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

    `// Evaluate the updated actors`
    Play one episode with actor π_{θ_j} and store all transitions in $\mathbb{R}$
    Get episode length T, discounted return R and state descriptors $\{\psi(s_1), \dots, \psi(s_T)\}$
    Store state descriptors $\{\psi(s_1), \dots, \psi(s_T)\}$ in $\mathbb{A}$
    Compute $\xi(\theta_j)$ and add the tuple $(R, \xi(\theta_j), \theta_j)$ in the MAP-Elites grid $\mathbb{M}$
    $actor_steps \leftarrow actor_steps + T$
  end

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

end

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. The actor is updated by minimizing the following loss function:

$$L^{actor}(\phi) = -\sum_{\text{batch}} Q_{\theta_1}(s_t, \pi_\phi(s_t)). \tag{7}$$

Policy evaluation and policy improvement steps are repeated until convergence. TD3 demonstrates state of the art performance on several MuJoCo benchmarks.

C QD-PG Details

C.1 Computational details

We consider populations of $N = 4$ actors for the POINT-MAZE environment 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 are run on a standard computer 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 MAP-ELITES grid which stores all solutions. An experiment on POINT-MAZE 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 GPU.

Computational costs of QD-PG mainly come from backpropagation during the update of each agent, and to the interaction between agents and the environment. These costs scale linearly with the population size but, as many other population-based methods, the structure of QD-PG lends itself very well to parallelization. We leverage this property and parallelize our implementation to assign one agent per CPU thread. Memory consumption also scales linearly with the number of agents. To reduce this consumption, we centralize the MAP-ELITES grid on a master worker and distribute data among workers when needed. With these implementation choices, QD-PG only needs a very accessible computational budget for all experiments.

C.2 MAP-Elites Implementation details

QD-PG uses a MAP-ELITES grid as archive of solutions. We assume that the BD space is bounded and can be discretized into an Cartesian grid. We discretize each dimension into m meshes, see Table 2 for the value of m depending on the environment. Hence, the number of cells in the MAP-ELITES grid equals m times the number of dimensions of the BD space. When a new solution θ is obtained after the mutation phase, we look for the cell corresponding to its BD, $\xi(\theta)$. If the cell is empty, the solution is added, otherwise the new solution replaces the solution already contained in the cell if its score $F(\theta)$ is better than the score of the already contained solution. During selection, we sample solutions uniformly from the MAP-ELITES grid.

C.3 Diversity reward computation

QD-PG optimizes solutions for quality but also for diversity at the state level. The diversity policy gradient updates the solutions so as to encourage them to visit states with novel state descriptors. The novelty of a state descriptor $\psi(s_t)$ is expressed through a diversity reward r_t^D. In practice, we maintain a FIFO archive $\mathbb{A}$ of the state descriptors encountered so far. When a transition $(s_t, a_t, r_t, s_{t+1}, \psi(s_t))$ is stored in the replay buffer, we also add $\psi(s_t)$ to $\mathbb{A}$. We only add a state descriptor in $\mathbb{A}$ if its mean Euclidean distance to its K nearest neighbors is greater than an acceptance threshold. This filtering step enables to keep the archive size reasonable and to facilitate the computation of the K nearest neighbors. The values of K and of the threshold are given in Table 2.

When a batch of transitions is collected during the update phase, we recompute fresh diversity rewards r_t^D as the mean Euclidean distance between the sampled state descriptors $\psi(s_t)$ and their K nearest neighbors in $\mathbb{A}$. These diversity rewards are used instead of standard rewards in sampled transitions $(s_t, a_t, r_t^D, s_{t+1}, \psi(s_t))$ to compute the diversity policy gradient.

C.4 QD-PG HYPER-PARAMETERS

Table 2 summarizes hyper-parameters used in experiments. Most of these hyper-parameter values are taken from TD3.

Table 2: QD-PG Hyper-parameters: ANT-MAZE and ANT-TRAP hyper-parameters are identical and grouped under the Ant column

Parameter	**PointMaze**	**Ant**
TD3		
Optimizer	SGD	SGD
Learning rate	6.10^{-3}	3.10^{-4}
Discount factor γ	0.99	0.99
Replay buffer size	10^6	5.10^5
Hidden layers size	$64/32$	$256/256$
Activations	ReLU	ReLU
Minibatch size	256	256
Target smoothing coeff.	0.005	0.005
Delay policy update	2	2
Target update interval	1	1
Gradient steps ratio	4	0.1
State Descriptors Archive		
Archive size	10000	10000
Threshold of acceptance	0.0001	0.1
K-nearest neighbors	10	10
MAP-Elites		
Nb. of bins per dimension	5	7

D ENVIRONMENTS

D.1 ENVIRONMENTS DETAILS

In POINT-MAZE an observation contains the agent position at time t, and an action corresponds to position increments along the x and y axes. The trajectory length cannot exceed 200 steps.

In ANT-MAZE the final performance is the maximum reward received during an episode. The environment is considered solved when an agent obtains a score superior to -10. An episode consists of 3000 time steps, this horizon is three times larger than in usual MUJOCO environments, making this environment particularly challenging for RL based methods (Vemula et al., 2019).

In ANT-TRAP, the trap consists of three walls forming a dead-end directly in front of the ant. In this environment, the trajectory length cannot exceed 1000 steps. In the three environments, the state descriptor is defined as the agent center of gravity position at time step t while the solution descriptor is the center of gravity position at the end of the trajectory.As opposed to POINT-MAZE and ANT-MAZE, where the objective is to reach the exit area, there is no unique way to solve ANT-TRAP and we expect a QD algorithm to generate various effective solutions as depicted in Figure 1. Furthermore, the behavior descriptor is not aligned with the fitness which makes it more difficult for

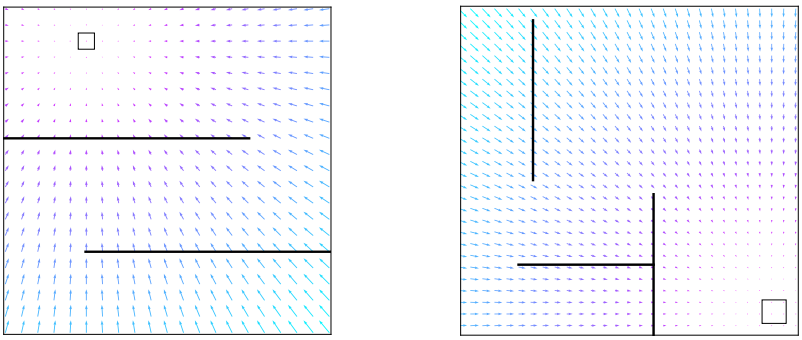
MAP-ELITES and also for pure novelty seeking approaches as simply exploring the BD space is not enough to find a performing solution.

D.2 ENVIRONMENTS ANALYSIS

In POINT-MAZE, the state and action spaces are two-dimensional. By contrast, in ANT-MAZE and ANT-TRAP, the dimensions of their observation spaces are respectively equal to 29 and 113 while the dimensions of their action spaces are both equal to 8, making these two environments much more challenging 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, unlike mazes where this space is restricted by the walls. In this case, a state descriptor corresponds to the ant position that is clipped to remain in a given range. On the y-axis, this range is defined as three times the width of the trap. 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 8b depicts the BD space in ANT-TRAP.

In all environments, state descriptors $\psi(s_t)$ are defined as the agent's position at time step t and behavior descriptors $\xi(\theta)$ are defined as the agent's position at the end of a trajectory. Therefore, we have $\mathcal{B} = \mathcal{D} = \mathbb{R}^2$, $\psi(s_t) = (x_t, y_t)$ and $\xi(\theta) = (x_T, y_T)$ where T is the trajectory length. We also take $||.||_{\mathcal{B}}$ and $||.||_{\mathcal{D}}$ as Euclidean distances. This choice does not always satisfy Equation equation 3 but is convenient in practice and led to satisfactory results. The peculiarity of ANT-TRAP lies in the fact that the reward is expressed as the forward velocity of the ant, thus making the descriptors not totally aligned with the task.

Figure 5 highlights the deceptive nature of the POINT-MAZE and the ANT-MAZE objective functions by depicting gradient fields in both environments. Similarly, the reward is also deceptive in ANT-TRAP.



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

Figure 5: Gradients maps on POINT-MAZE and ANT-MAZE. Black lines are maze walls, arrows depict gradient fields and the square indicates the maze exit. Both settings present deceptive gradients as naively following them leads into a wall.

E DETAILED RESULTS

In this section, we provide performance charts corresponding to Table 1b and Table 1a, coverage maps highlighting the exploration capabilities of QD-PG, and detailed results of the fast adaptation experiment. Table 3 summarizes the different components present in QD-PG, its ablations and all baselines.

E.1 PERFORMANCE CHARTS

Figure 7 shows supplementary results, with an ablation study and a comparison to evolutionary competitors in ANT-MAZE. In QD-PG, the current population of solutions is evaluated every 150.000 time steps in ANT-MAZE and ANT-TRAP, and every 5000 time steps in POINT-MAZE. At evaluation time, agents are set to be deterministic and stop exploring. Figure 7 reports the performance obtained by the best agent in the population at a given time step.

Table 3: Ablations and baselines summary. Selec. stands for selection. The last column assesses whether the method optimizes for a collection instead of a single solution.

	Algorithm	QPG	DPG	Q Selec.	D Selec.	Collection
Ablations	QD-PG	✓	✓	✓	✓	✓
	QD-PG SUM	✓	✓	✓	✓	✓
	D-PG	X	✓	✓	✓	✓
	Q-PG	✓	X	✓	✓	✓
PG	SAC	✓	X	X	X	X
	TD3	✓	X	X	X	X
	RND	✓	✓	X	X	X
	CEM-RL	✓	X	✓	X	✓
	P3S-TD3	✓	✓	X	X	✓
	AGAC	✓	✓	X	X	X
	DIAYN	X	✓	X	X	X
	PGA-ME	✓	X	✓	✓	✓
QD	ME-ES	X	X	✓	✓	✓
	NSR-ES	X	X	✓	✓	✓
	NSRA-ES	X	X	✓	✓	✓

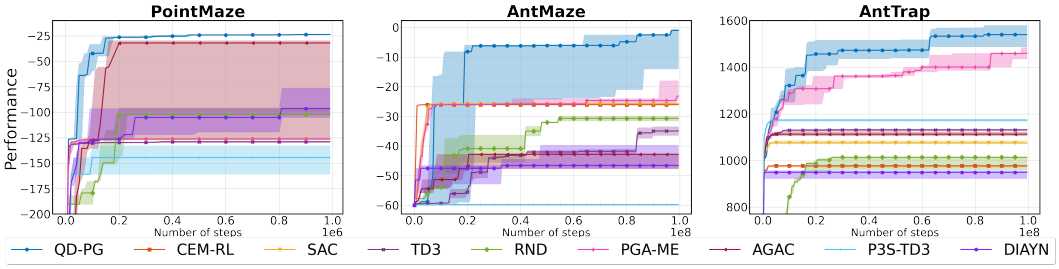
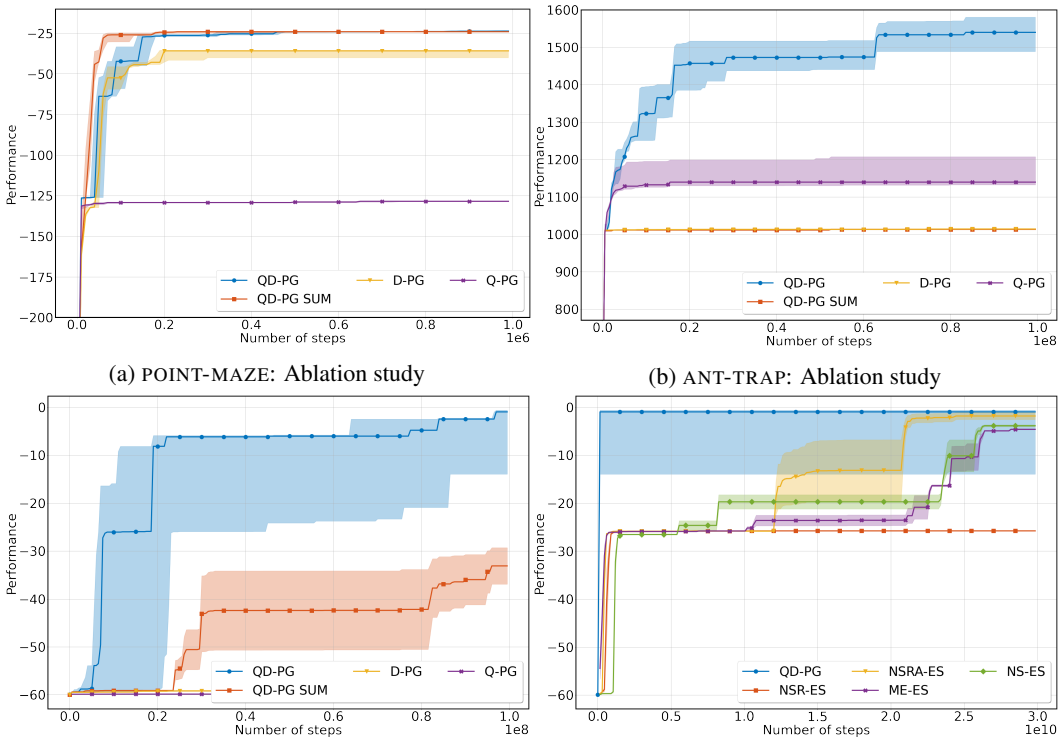


Figure 6: Performance of QD-PG and baseline algorithms for POINT-MAZE (10^6 steps), ANT-MAZE (10^8 steps) and ANT-TRAP (10^8 steps). Plots present median bounded by first and third quartiles.

E.2 ABLATION STUDY

In addition to Section 7, we wanted to answer the following question: What is the usefulness of the different components of QD-PG? To answer this question, we propose to investigate the following matters: Can we replace alternating quality and diversity updates by a single update that optimizes for the sum of both criteria? Are quality (resp. diversity) gradients updates alone enough to fill the MAP-ELITES grid? Consequently, we consider the following ablations of QD-PG: QD-PG SUM computes a gradient to optimize the sum of the quality and diversity rewards, D-PG applies only diversity gradients to the solutions, and Q-PG applies only quality gradients, but both D-PG and Q-PG still use QD selection.

The ablation study in Table 1a and Figure 7 shows that when maximising quality only, Q-PG fails due to the deceptive nature of the reward and when maximizing diversity only, D-PG sufficiently explores to solve POINT-MAZE but requires more steps and finds lower-performing solutions. When optimizing simultaneously for quality and diversity, QD-PG SUM fails to learn in ANT-TRAP and manages to solve the task in ANT-MAZE but requires more samples than QD-PG. We hypothesize that quality and diversity rewards may give rise to conflicting gradients. For instance, at the beginning of training in ANT-TRAP, the quality reward drives the ant forward whereas the diversity reward drives it back to escape the trap and explore the environment. Therefore, both rewards cancel each other, preventing any learning. This study validates the usefulness of QD-PG components: 1) optimizing for diversity is required to overcome the deceptive nature of the reward; 2) adding quality optimization provides better asymptotic performance; 3) it is better to disentangle quality and diversity updates.



(a) POINT-MAZE: Ablation study

(b) ANT-TRAP: Ablation study

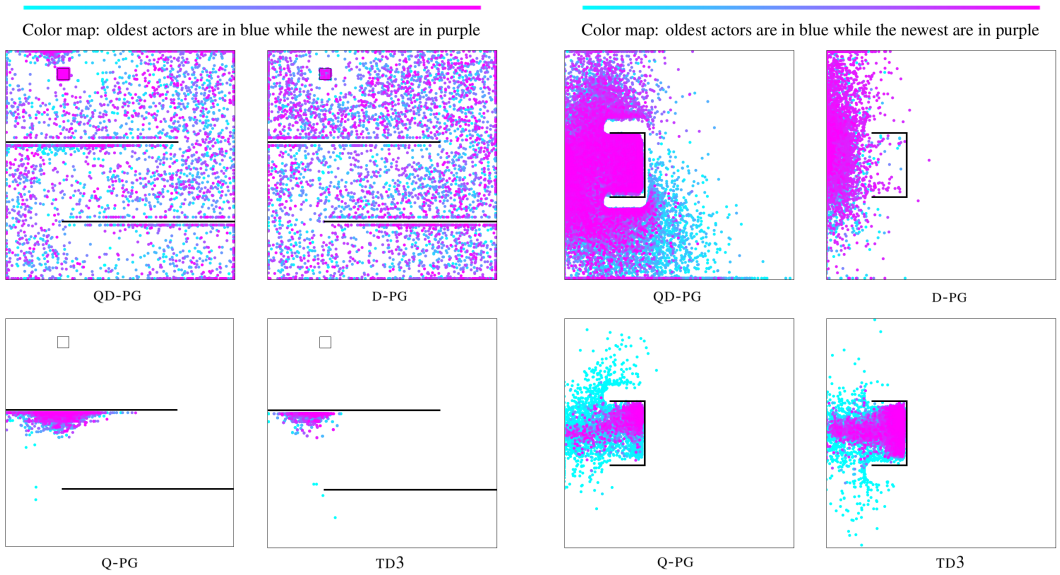
(c) ANT-MAZE: Ablation study

(d) ANT-MAZE: Evo methods

Figure 7: Learning curves of QD-PG versus ablations and evolutionary baselines. In POINT-MAZE and ANT-TRAP, the performance is the highest return. In ANT-MAZE, it is the negative lowest distance to the goal. We separate the comparison on ANT-MAZE into two graphs for better readability. Plots present median bounded by first and third quartiles.

E.3 COVERAGE MAPS

Figure 8a shows coverage maps of the POINT-MAZE environment obtained with one representative seed by the different algorithms presented in the ablation study (see Table 1a). 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.



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

Figure 8: Coverage map of the POINT-MAZE and ANT-TRAP environments for all ablations. Each dot corresponds to the final position of an agent.

QD-PG and D-PG almost cover the whole BD space including the objective. Unsurprisingly, Q-PG and TD3 present very poor coverage maps, both algorithms optimize only for quality and the MAP-ELITES selection mechanism in Q-PG contributes nothing in this setting. By contrast, algorithms optimizing for diversity (QD-PG and D-PG) find the maze exit. However, as shown in Table 1a, QD-PG which also optimizes for quality, is able to refine trajectories through the maze and obtains significantly better performance.

Figure 8b depicts the coverage maps of the ANT-TRAP environment by QD-PG and TD3. Only QD-PG is able to bypass the trap and to cover a large part of the BD space.

E.4 FAST ADAPTATION

The fast adaptation experiment described in Section 7 uses a Bayesian optimization process to quickly find a high-performing solution for a new randomly sampled goal. Browsing the MAP-ELITES grid in an exhaustive way is another option to find a good solution for a new objective. However, the number of solutions to be tested with this option increases quadratically w.r.t. the number of meshes used to discretize the dimensions of the BD space. As shown in Table 2, we use a 7×7 grid to train QD-PG in the ANT-MAZE environment, containing a maximum of 49 solutions. In this setting, the difference in computation cost between exhaustive search and Bayesian optimization is negligible.

To ensure that fast adaptation scales to finely discretized MAP-ELITES grids, we reproduce this experiment with a 100×100 grid, thus containing thousands of solutions. We first train QD-PG again on the standard objective of ANT-MAZE and obtain a 100×100 grid of solutions. Then, we repeat the fast adaptation experiment described in Section 7 using this large grid. With a budget of only 50 solutions to be tested during the Bayesian optimization process among the thousands of solutions contained in the grid, we are able to recover a good solution for the new objective.

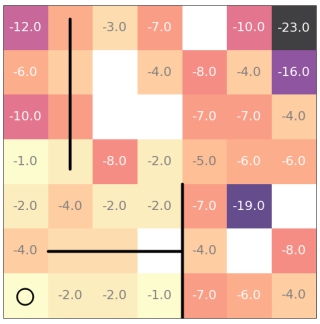


Figure 9: Performance map of 100 fast adaptation experiments in ANT-MAZE. In each square, we display the score of the best experiment whose goal was sampled in this region of the maze, as several experiments may have goals in the same square. White squares correspond to regions where no goal was sampled during the experiments. The black circle shows the agent's starting position.

We repeat this experiment 100 times, each time with a new random goal, and obtain an average performance of -9 with a standard deviation of 7.

Figure 9 maps these 100 fast adaptation experiments to their respective goal location and performance. In each square, we display the score of the best experiment whose goal was sampled in this region of the maze. For instance, the square in the top left corner of the performance map corresponds to one of the 100 fast adaptation experiments that sampled its goal in this part of the maze, and obtained a performance of -12 after testing 50 solutions from the MAP-ELITES grid during the Bayesian optimization process. Some squares do not have a score when no experiment sampled its goal in this region of the maze.

F D-PG DERIVATIONS EXTENDED

F.1 PROOF OF PROPOSITION 1.

The *diversity* of a set of K solutions $\{\theta_k\}_{k=1,\dots,K}$ is defined as $d : \Theta^K \to \mathbb{R}^+$:

$$d\left(\{\theta_k\}_{k=1,\dots,K}\right) = \sum_{i=1}^{K} \min_{k \neq i} ||\xi(\theta_i), \xi(\theta_k)||_{\mathcal{B}}, \tag{8}$$

The diversity can be split into three terms: the distance of θ_1 to its nearest neighbor (defined as θ_2), the distance of θ_1 to the θ_j for which θ_1 is the nearest neighbor (defined $\{\theta_j\}_{j=3,\dots,K}$ [1]) and a third term that does not depend on θ_1. Thus:

$$\begin{aligned} d(\{\theta_k\}_{k=1,\dots,K}) &= ||\xi(\theta_1), \xi(\theta_2)||_{\mathcal{B}} + \sum\nolimits_{j=3}^{J} ||\xi(\theta_1), \xi(\theta_j)||_{\mathcal{B}} + M \\ &= \sum\nolimits_{j=2}^{J} ||\xi(\theta_1), \xi(\theta_j)||_{\mathcal{B}} + M \end{aligned}$$

where $M = \sum_{i \notin \{1,\dots,J\}} \min_{k \neq i} ||\xi(\theta_i), \xi(\theta_k)||_{\mathcal{B}}$ does not depend on θ_1. Hence, the gradient with respect to θ_1 is:

$$\nabla_{\theta_1} d(\{\theta_k\}_{k=1,\dots,K}) = \nabla_{\theta_1} \sum\nolimits_{j=2}^{J} ||\xi(\theta_1), \xi(\theta_j)||_{\mathcal{B}}$$

As the remaining term is precisely defined as the novelty n :

$$n(\theta_1, (\theta_j)_{2 \leq j \leq J}) = \sum\nolimits_{j=2}^{J} ||\xi(\theta_1), \xi(\theta_j)||_{\mathcal{B}} \tag{9}$$

[1] Remark: θ_2 can appear twice in the list $(\theta_j)_{2 \leq j \leq J}$

We get the final relation :

$$\nabla_{\theta_1} d(\{\theta_k\}_{k=1,\dots,K}) = \nabla_{\theta_1} n(\theta_1, (\theta_j)_{2\leq j\leq J}) \tag{10}$$

F.2 Motivation behind Equation 3

The idea behind this equation is to link novelty defined at the solution level to a notion of novelty defined at the time step level. The information that we use at the time step level is the current state in the environment, so we describe a solution by the states it visits and hence want to define state novelty such that :

$$n(\theta_1, (\theta_j)_{2\leq j\leq J}) = \mathbb{E}_{\pi_{\theta_1}} \sum_t n(s_t, (\theta_j)_{2\leq j\leq J}) \tag{11}$$

Furthermore, we assume that a state is novel w.r.t. some solutions if it is novel w.r.t. to the states visited by these solutions. To be able to do so, we introduce the notion of state descriptor Ψ, which enables to define the novelty between states. Hence, we can define the novelty of a state w.r.t. to a set of solutions by:

$$n(s, (\theta_j)_{j=1,\dots,J}) = \sum_{j=1}^{J} \mathbb{E}_{\pi_{\theta_j}} \sum_t ||\psi(s), \psi(s_t)||_{\mathcal{D}} \tag{12}$$

This state descriptor Ψ constrains the link between the novelty at the state level with the novelty at the solution level through a link with the behavior descriptor ξ.

That being said, we can now compute the diversity gradient thanks to the novelty at the state level. As a matter of fact, proposition 1 links diversity gradient to novelty at solution level:

$$\nabla_{\theta_1} d(\{\theta_k\}_{k=1,\dots,K}) = \nabla_{\theta_1} n(\theta_1, (\theta_j)_{2\leq j\leq J}) \tag{13}$$

Then we link novelty at the solution level to novelty at the state level with the following equation, which is satisfied thanks to the relevant choice of the state descriptor Ψ.

$$n(\theta_1, (\theta_j)_{2\leq j\leq J}) = \mathbb{E}_{\pi_{\theta_1}} \sum_t n(s_t, (\theta_j)_{2\leq j\leq J}) \tag{14}$$

Finally, by replacing the novelty at the solution level by the novelty at the state level in proposition 1, we get the formulation of the diversity policy gradient given in equation 4.

$$\nabla_{\theta_1} d(\{\theta_k\}_{k=1,\dots,K}) = \nabla_{\theta_1} \mathbb{E}_{\pi_{\theta_1}} \sum_t n(s_t, (\theta_j)_{2\leq j\leq J}) \tag{15}$$