EVOLVING POPULATIONS OF DIVERSE RL AGENTS WITH MAP-ELITES

Anonymous authors
Paper under double-blind review

ABSTRACT

Quality Diversity (QD) has emerged as a powerful alternative optimization paradigm that aims at generating and maintaining large and diverse collections of solutions, notably with its flagship algorithm MAP-ELITES (ME) which evolves solutions through mutations and crossovers. While very effective for some unstructured problems, early ME implementations relied exclusively on random search to evolve the population of solutions, rendering them notoriously sample-inefficient for high-dimensional problems, such as when evolving neural networks. Follow-up works considered exploiting gradient information to guide the search in order to address these shortcomings through techniques borrowed from either Black-Box Optimization (BBO) or Reinforcement Learning (RL). While mixing RL techniques with ME unlocked state-of-the-art performance for robotics control problems that require a good amount of exploration, it also plagued these ME variants with limitations common among RL algorithms that ME was so far free of, such as hyperparameter sensitivity, high stochasticity as well as training instability, including when the population size increases as some components are shared across the population in recent approaches. Furthermore, existing approaches mixing ME with RL tend to be tied to a specific RL algorithm, which effectively prevents their use on problems where the corresponding RL algorithm fails. To address these shortcomings, we introduce a flexible framework that allows the use of any RL algorithm and alleviates the aforementioned limitations by evolving populations of agents (whose definition include hyperparameters and all learnable parameters) instead of just policies. We demonstrate the benefits brought about by our framework through extensive numerical experiments on a number of robotics control problems, some of which with deceptive rewards, taken from the QD-RL literature. We also open source an efficient JAX-based implementation of our algorithm.

1 INTRODUCTION

Drawing inspiration from natural evolution's ability to produce living organisms that are both diverse and high-performing through competition in different niches, Quality Diversity (QD) methods evolve populations of diverse solutions to solve an optimization problem. In contrast to traditional Optimization Theory, where the goal is to find one solution maximizing a given scoring function, QD methods explicitly use a mapping from solutions to a vector space, referred to as a *behavior descriptor space*, to characterize solutions and maintain a data structure, referred to as a *repertoire*, filled with high-performing solutions that cover this space as much as possible, in a process commonly referred to as *illumination*. This new paradigm has led to breakthroughs over the past decade in many domains ranging from robotics control to engineering design and games generation Gaier et al. (2018); Sarkar & Cooper (2021); Gravina et al. (2019); Cully & Demiris (2018). There are a number of advantages to QD methods over standard optimization ones. Actively seeking and maintaining diversity in a population of solutions has proved to be an effective exploration strategy, by reaching high-performing regions through a series of stepping stones, when the fitness function has no particular structure Gaier et al. (2019). Additionally, having at disposal a diverse set of high-performing solutions can be greatly beneficial to a decision maker Lehman et al. (2020), for instance because the scoring function may fail to model accurately the reality Cully et al. (2015).

MAP-ELITES Mouret & Clune (2015) has emerged in the past few years as one of the most widely used algorithm in the QD community, notably for its simplicity and efficacy. MAP-ELITES divides

the behavior descriptor space into a discrete mesh of cells and strives to populate each cell with a solution with a matching behavior descriptor that maximizes the fitness function as much as possible. This algorithm has been used in many practical applications with great success, such as developing closed-loop controllers for hexapod robots that can adapt to damage in real time Cully et al. (2015). However, just like many evolutionary algorithms, MAP-ELITES struggles on problems with high-dimensional search spaces, such as when evolving controllers parametrized by neural networks, as it relies on random mutations and crossovers to evolve the population.

The breakthroughs of Deep Reinforcement Learning in sequential decision making problems prompted a new line of work in the QD field to make the algorithms capable of dealing with deep neural network parametrizations. These new methods borrow techniques from either Black-Box Optimization (BBO) or Reinforcement Learning (RL) in order to exploit gradient information to guide the search. Methods based on BBO techniques Colas et al. (2020); Conti et al. (2018) follow the approaches from earlier works on scaling evolutionary algorithms to neuro-evolution, such as Salimans et al. (2017); Stanley & Miikkulainen (2002), and empirically evaluate gradients w.r.t. the parameters by stochastically perturbing them by small values a number of times. Methods borrowing tools from RL, such as Nilsson & Cully (2021); Pierrot et al. (2022), exploit the Markov-Decision-Process structure of the problem and adapt off-policy RL algorithms, such as TD3 Fujimoto et al. (2018), to evolve the population. This often entails adding additional components to the evolutionary algorithm (e.g. a replay buffer, critic networks, hyperparameters of the RL agent, ...) and methods differ along the way these components are managed. RL-based MAP-ELITES approaches have outperformed other MAP-ELITES variants, and even state-of-the art RL methods, on a variety of robotics control problems that require a substantial amount of exploration due to deceptive or sparse rewards. However, the introduction of RL components in MAP-ELITES has come with a number of downsides: (i) high sensibility to hyperparameters Khadka et al. (2019); Zhang et al. (2021), (ii) training instability, (iii) high variability in performance, and perhaps most importantly (iv) limited parallelizability of the methods due to the fact that many components are shared in these methods for improved sample-efficiency. Furthermore, existing RL-based MAP-ELITES approaches are inflexibly tied to a specific RL algorithm, which effectively prevents their use on problems where the latter fails.

These newly-introduced downsides are particularly problematic as they are some of the main advantages offered by evolutionary methods that are responsible for their widespread use. These methods are notoriously trivial to parallelize and there is almost a linear scaling between the convergence speed and the amount of computational power available, as shown in Lim et al. (2022) for MAP-ELITES. This is all the more relevant with the advent of modern libraries, such as JAX Bradbury et al. (2018), that seamlessly enable not only to distribute the computations, including computations taking place in the physics engine with BRAX Freeman et al. (2021), over multiple accelerators but also to fully leverage their parallelization capabilities through automated vectorization primitives, see Lim et al. (2022); Flajolet et al. (2022); Tang et al. (2022). Evolutionary methods are also notoriously robust to the exact choice of hyperparameters, see Khadka et al. (2019), which makes them suited to tackle new problems. This is in stark contrast with RL algorithms that tend to require problem-specific hyperparameter tuning to perform well Khadka et al. (2019); Zhang et al. (2021).

In this work, in order to overcome the aforementioned limitations of RL-based MAP-ELITES approaches, we develop a new MAP-ELITES framework that **1.** can be generically and seamlessly compounded with any RL agent, **2.** is robust to the exact choice of hyperparameters by embedding a meta-learning loop within MAP-ELITES, **3.** can scale to large population sizes, which helps alleviating stochasticity and training stability issues, without entering offline RL regimes by independently evolving populations of entire agents (which includes all of their components, such as replay buffers, critic networks, ...) instead of evolving policies only and sharing the other components across the population. Our method, dubbed PBT-MAP-ELITES, builds on MAP-ELITES and combines standard isoline operators with policy gradient updates to get the best of both worlds. We evaluate PBT-MAP-ELITES when used with the Soft Actor Critic SAC Haarnoja et al. (2018) and TD3 Fujimoto et al. (2018) agents on a set of five standard robotics control problems taken from the QD literature and show that it either yields performance on par with or outperforms state-of-the-art MAP-ELITES approaches, in some cases by a strong margin, while not being provided with hyperparameters tuned beforehand for these problems. Finally, we open source an efficient JAX-based implementation of our algorithm that combines the efficient implementation of PBT from Flajolet et al. (2022) with that of MAP-ELITES from Lim et al. (2022). We refer to these two prior works for speed-up data points compared to alternative implementations.

2 BACKGROUND

Problem Definition. We consider the problem of generating a repertoire of neural policies that are all high-performing for a given task while maximizing the diversity of policies stored in the repertoire. More formally, we consider a finite-horizon Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$, where $\mathcal{A}$ is the action space, $\mathcal{S}$ is the state space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward signal, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, and T is the episode length. A neural policy corresponds to a neural network $\pi_\theta : \mathcal{S} \to \mathcal{D}(\mathcal{A})$ where $\theta \in \Theta$ denotes the weights of the neural network and $\mathcal{D}(\mathcal{A})$ is the space of distributions over the action space. At each time step, we feed the current environment state to the neural network and we sample an action from the returned distribution, which we subsequently take. Once the action is carried out in the environment, we receive a reward and the environment transitions to a new state. The fitness $F(\pi_\theta)$ of a policy π_θ is defined as the expected value of the sum of rewards thus collected during an episode. We denote the space of trajectories thus followed in the environment by $\tau \in \Omega$. In the QD literature, diversity is not directly measured in the parameter space Θ, but rather in another space $\mathcal{D}$, referred to as the behavior descriptor space or sometimes simply descriptors space, which is defined indirectly through a pre-specified and problem-dependent mapping $\Phi : \Omega \to \mathcal{D}$. A policy π_θ is thus characterized by rolling it out in the environment and feeding the trajectory to Φ. With a slight abuse of notation, we denote by $\Phi(\pi_\theta)$ the behavior descriptor of the policy π_θ. Diversity of a repertoire of policies is measured differently across QD approaches.

MAP-Elites. MAP-ELITES uses a tesselation technique to divide the descriptor space into a finite number of cells, which collectively define a discrete repertoire. In this work, we use the Centroidal Voronoi Tessellation (CVT) technique Vassiliades et al. (2017) for all considered methods it has been shown to be general and easy to use in practice Vassiliades et al. (2017); Pierrot et al. (2022). MAP-ELITES starts by randomly initializing a set of M policies. Each of these policies is then independently evaluated in the environment and they are sequentially inserted into the repertoire according to the following rule. If the cell corresponding to the descriptor of the policy at hand is empty, the policy is copied into this cell. In the opposite situation, the policy replaces the current incumbent only if it has a greater fitness and is dropped otherwise. During each subsequent iteration, policies are randomly sampled from the repertoire, copied, and perturbed to obtain a new set of M policies which are then tentatively inserted into the repertoire following the aforementioned rule. Implementations of MAP-ELITES often differ along the exact recipe used to perturb the policies. The original MAP-ELITES algorithm Mouret & Clune (2015) relies on random perturbations. In this work, we use the isoline variation operator Vassiliades & Mouret (2018) that, given two parent policies, say policies θ_1 and θ_2, adds Gaussian noise $\mathcal{N}(0, \sigma_1)$ to θ_1 and offsets the results along the line $\theta_2 - \theta_1$ by a magnitude randomly sampled from a zero-mean Gaussian distribution with variance $\mathcal{N}(0, \sigma_2)$. This strategy has proved to be particularly effective to evolve neural networks Rakicevic et al. (2021). Pseudocode for MAP-ELITES is provided in the Appendix.

BBO-based QD. To improve sample efficiency and asymptotic performance, methods such as ME-ES Colas et al. (2020) use first-order updates to perturb the policies with the objective of both increasing the fitness of the policies in the repertoire and improving the coverage of the repertoire (i.e. the number of non-empty cells). To generate the updates, ME-ES use the Evolution Strategy introduced in Salimans et al. (2017). Specifically, after selecting a policy from the repertoire, its neural network parameters are perturbed stochastically with a small amount of Gaussian noise a number of times and the resulting policies are rolled out in the environment for a full episode. All of the collected samples are then used to empirically estimate gradients for a smoothed version around the starting policy of either (1) the fitness function, (2) a novelty function which is defined as the average Euclidean distance between the starting policy's behavior descriptor and its k nearest neighbors among all previously computed behavior descriptors, or (3) alternatively the fitness function and the novelty function to increase both quality and diversity, which is the version we use in this work (see the Appendix for the pseudocode). Note that similar strategies using the NS-ES family of algorithms exist, such as Conti et al. (2018), but these methods are outperformed by ME-ES Colas et al. (2020).

RL-based QD. Using evolution strategies to guide the search with first-order updates improves upon random search but remains doomed to a low sample-efficiency due to the need of rolling out a significant number of policies over entire trajectories to get reasonably accurate gradient estimates. More recent techniques, such as QD-PG Pierrot et al. (2022) and PGA-MAP-ELITES Nilsson & Cully (2021), exploit the MDP structure of the problem and leverage policy-gradient techniques from RL as well as off-policy extensions for improved sample efficiency and better asymptotic convergence.

Both QD-PG and PGA-MAP-ELITES build on the TD3 agent Fujimoto et al. (2018). PGA-MAP-ELITES combines random mutations derived through the isoline variation operator with mutations obtained through policy gradient computations. QD-PG introduces the notion of a diversity reward, a signal defined at the timestep-level to drive policies towards unexplored regions of the behavior descriptor space, which makes it possible to leverage the RL machinery to compute policy gradients to increase the diversity of the population, referred to as diversity policy gradients, in addition to the standard policy gradients to increase the fitness of the policies, referred to as quality policy gradients. At each MAP-ELITES iteration, half of the selected policies are updated using quality policy gradients and the other half are updated using diversity policy gradients. In contrast to PGA-MAP-ELITES, QD-PG does not relies on random search updates. Both QD-PG and PGA-MAP-ELITES use a single shared replay buffer where all the transitions collected when evaluating the agents are stored and from which batches are sampled to compute policy gradients.

Critic networks are managed differently by each algorithm. QD-PG uses two different sets of critic parameters, one for quality rewards and one for diversity rewards, that are shared across the population and both are updated any time a policy gradient is computed. PGA-MAP-ELITES maintains a greedy policy and its associated critic which are updated independently of the rest of the repertoire. The greedy policy is regularly inserted in the repertoire and the critic is used to compute policy gradients updates for all other policies but is only updated using the greedy policy.

These precise design choices not only make PGA-MAP-ELITES and QD-PG difficult to distribute efficiently but they also harm the flexibility of these methods. For instance, if one would like to replace TD3 by another popular off-policy algorithm such as SAC, which is known to perform better for some environments, numerous new design choices arise. For instance for SAC, one would have to decide how to handle the temperature parameter and the entropy target within the population. Furthermore, while sharing critic parameters and using a single replay buffer was motivated by a desire for greater sample efficiency, this introduces new issues when scaling these methods. For instance, as the number of policies updated concurrently at each iteration increases we get closer to an offline RL setting, which is known to harm performance, since all policies share the same replay buffer. Conversely, as the size of the repertoire increases, any single policy stored in the repertoire is updated all the less frequently than the critic which may cause them to significantly lag behind over time. Finally, both QD-PG and PGA-MAP-ELITES assume that good hyperparameters are provided for TD3 while it is known that tuning these values for the problem at hand is necessary to get good performance. This effectively puts the burden on the user to tune hyperparameters for TD3 as a preliminary step, which limits the usability of such methods in new settings. Pseudocodes for QD-PG and PGA-MAP-ELITES are provided in the Appendix.

3 METHOD

In order to overcome the limitations of RL-based QD methods identified in the last section, we revisit the neuro-evolution problem defined in Section 2 and introduce a new algorithm, dubbed PBT-MAP-ELITES, that evolves populations of agents as opposed to populations of policies. An agent is defined by a tuple $(\theta, \phi, \mathbf{h})$ where θ denotes the policy parameters, ϕ denotes all other learnable parameters of the agent (e.g. critic parameters and target critic parameters), and $\mathbf{h}$ denotes its hyperparameters (e.g. learning rates and magnitude of the exploration noise). As in the original formulation, we assume that the fitness and behavior descriptor functions depend only on the policy, i.e. on θ. The learnable parameters and the hyperparameters are only used when agents are updated. PBT-MAP-ELITES internally uses a policy-search-based RL algorithm which can be selected freely by the user. In particular, it may be on-policy or off-policy.

PBT-MAP-ELITES maintains a MAP-ELITES repertoire as well as a population of P agents. The population is randomly initialized (including the hyperparameters), evaluated, copied and inserted into the repertoire. We also initialize P replay buffers if the underlying RL algorithm makes use of them. Additionally, a batch of agents is sampled from the repertoire and a variation operator is applied to obtain M offspring that are also evaluated and inserted into the repertoire as part of the initialization phase. Then, the algorithm proceeds in iterations, each of which consists of two consecutive steps: 1. population update and 2. MAP-ELITES repertoire update.

Population Update. To update the population of agents, we use the following strategy inspired from Jaderberg et al. (2017). We first rank all agents in the population by fitness based on the evaluation

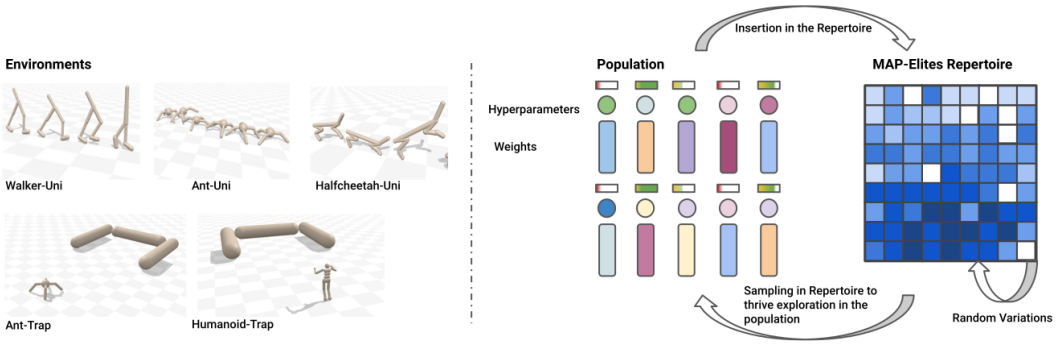

Figure 1: **Left Panel.** The set of QD-RL environments used to evaluate all methods. All environments are implemented using the BRAX simulator and are available in the QDAX suite. In WALKER2D-UNI, ANT-UNI, and HALFCHEETAH-UNI, the goal is to learn to control the corresponding robot to run as fast as possible with gaits that are as diverse as possible. In ANT-TRAP and HUMANOID-TRAP, the robot should travel as far as possible along the x-axis direction, which entails sidestepping the trap to get the highest possible return thus putting the exploration capabilities of considered methods to test. **Right panel.** Illustration of the PBT-MAP-ELITES algorithm. PBT-MAP-ELITES maintains a population of agents (policy parameters, other learnable parameters as well as hyperparameters) and a MAP-ELITES repertoire of agents. The repertoire update, generating more diversity, and the population update, improving the fitness of the policies, are intertwined.

that took place at the end of the last iteration. Agents that are in the bottom $p\%$ of the population are replaced by agents sampled uniformly from the top $n\%$ of the population, with $0 < p < 1 - n < 1$. We also randomly select $k\%$ of the agents in the population among the ones that are neither in the top $n\%$ nor in the bottom $p\%$ and we replace them by agents randomly sampled from the current MAP-ELITES repertoire. All other agents remain unchanged. This mechanism allows potentially lower-performing, but more diverse, individuals from the repertoire to enter the population while maintaining high-performing agents alive. When agents are replaced, new hyperparameter values are sampled uniformly from pre-specified ranges. The agents' policy parameters as well as all other learnable parameters are subsequently trained for S steps, using the user-selected RL algorithm. If needed, the collected experience is stored inside the replay buffers. In contrast to PGA-MAP-ELITES and QD-PG, we closely follow the general recipe followed by most RL algorithms and only add the experience collected during training, in exploration mode, to the replay buffers while the experience collected during evaluation, in exploitation mode, is discarded. Additionnaly, note that the agents are trained independently from one another, which makes it trivial to parallelize the most computationally intensive part of this step. This is in stark contrast with other MAP-ELITES-RL methods that share some parameters across the population, e.g. the critic parameters for QD-PG and PGA-MAP-ELITES, which are typically updated concurrently by all agents.

Repertoire Update. Once the agents in the population have been trained, they are evaluated and inserted into the repertoire. Then, just like during the initialization phase, a batch of agents is randomly sampled from the repertoire and undergoes a variation operator to obtain M offspring which are evaluated and inserted into the grid. As in PGA-MAP-ELITES, the variation operator is meant to increase the descriptor space coverage but we have also observed that this process stabilizes the algorithm as a whole. In order to define a variation operator that can be used with agents, as opposed to policies, we deal with variations over the policy and learnable parameters separately from variations over the hyperparameters. Specifically, an isoline operator is applied to policy and other learnable parameters while the offspring simply inherit the hyperparameters of one of their parents. While more sophisticated strategies could be investigated, we have observed that this simple mechanism works well in practice in our experiments.

Observe that optimization of the quality as well as the diversity of the policies happens at two different levels in PBT-MAP-ELITES. Quality is encouraged through both the elitist population update and the repertoire insertion mechanism. Diversity is induced through both the addition of agents from the repertoire to the population and the use of random variation operators at each iteration. The pseudocode of the algorithm is provided in the Appendix.

4 LITERATURE REVIEW

Quality Diversity. QD methods aim to simultaneously maximize diversity and performance. Among existing options, MAP-ELITES and Novelty Search with Local Competition (NSLC) are two of the most popular QD algorithms. NSLC builds on the Novelty Search algorithm Lehman & Stanley (2011) and maintains an unstructured archive of solutions selected for their high performance relative to other solutions in their neighborhoods while MAP-ELITES relies on a tesselation technique to discretize the descriptor space into cells. Both algorithms rely extensively on Genetic Algorithms (GA) to evolve solutions. As a result, they struggle when the dimension of the search space increases, which limits their applicability. These approaches have since been extended using tools from Evolution Strategies (ES) to improve sample efficiency and asymptotic performance over the original implementations based on GA Salimans et al. (2017). CMA-MAP-ELITES Fontaine et al. (2020) relies on Covariance Matrix Adaptation (CMA) to speed up the illumination of the descriptor space. NSRA-ES and NSR-ES Conti et al. (2018) build on recent ES tools to improve QD methods' exploration capabilities on deep RL problems with deceptive or sparse rewards. ME-ES Colas et al. (2020) introduces alternate ES updates for quality and diversity in order to solve deep RL problems with continuous action spaces that require a good amount of exploration. While ES-based approaches improve over GA-based ones, they are still relatively sample-inefficient due to the fact that they need to roll out a large of number of policies over entire trajectories to empirically estimate gradients with reasonable accuracy. Several recent methods propose to exploit analytical gradients when this is possible instead of estimating them empirically. DQD Fontaine & Nikolaidis (2021) builds a mutation operator that first computes gradients of the fitness and behavior descriptor functions at the current solution and carry out a first-order step by summing the gradients with random coefficients. Tjanaka et al. (2022) applies the same technique to deep RL problems with continuous action spaces. PGA-MAP-ELITES Nilsson & Cully (2021) and QD-PG Pierrot et al. (2022) exploit the MDP structure of the problems to compute policy gradients using the TD3 algorithm, outperforming all QD competitors for deep RL problems with continuous actions. However, both methods are tied a single RL algorithm and are highly sensible to the choice of hyperpameters for the TD3 algorithm.

Population Based Reinforcement Learning. Our work has different motivations than classical RL algorithms as we do not aim to find a policy than achieves the best possible return but rather to illuminate a target descriptor space. However, we share common techniques with Population-Based RL (PBRL) algorithms. In this field, the closest method to ours is the Population-Based-Training (PBT) algorithm Jaderberg et al. (2017) which uses a genetic algorithm to learn the hyperparameters of a population of RL agents concurrently to training them. While PBT-MAP-ELITES and PBT use similar strategies to update the population of agents, PBT only seeks the highest-performing agent by extracting the best one from the final population while PBT-MAP-ELITES aims to find a diverse collection of high-performing agents. Several methods such as CERL, ERL, and CEM-RL Pourchot & Sigaud (2019); Khadka & Tumer (2018); Khadka et al. (2019) combine ES algorithms with PBRL methods to improve the asymptotic performance and sample efficiency of standard RL methods. Other methods, such as DvD Parker-Holder et al. (2020) and P3S-TD3 Jung et al. (2020), train populations of agents and add terms in their loss functions to encourage the agents to explore different regions of the state-action space but always with the end goal of maximizing the performance of the best agent in the population. Flajolet et al. (2022) show how to vectorize computations across the population to run PBRL algorithms as efficiently as possible on accelerators through the use of the JAX library. Lim et al. (2022) introduced similar techniques to accelerate MAP-ELITES through the evaluation of thousands of solutions in parallel with JAX. In this study, we build on both of these works and implement PBT-MAP-ELITES in the JAX framework to make it fast and scalable.

5 EXPERIMENTS

Environments. We use five robotics environments that fall into two categories:

1. HALFCHEETAH-UNI, WALKER2D-UNI and ANT-UNI are environments widely used in the QD community to evaluate an algorithm's ability to illuminate a complex descriptor space, see for instance Cully et al. (2015); Nilsson & Cully (2021); Tjanaka et al. (2022). In these environments, the goal is to make a legged robot run as fast as possible along the forward direction while optimizing for diversity w.r.t. the robot's gaits, indirectly characterized as the mean frequencies of contacts between the legs and the ground. This last quantity defines the behavior descriptor for these envi-

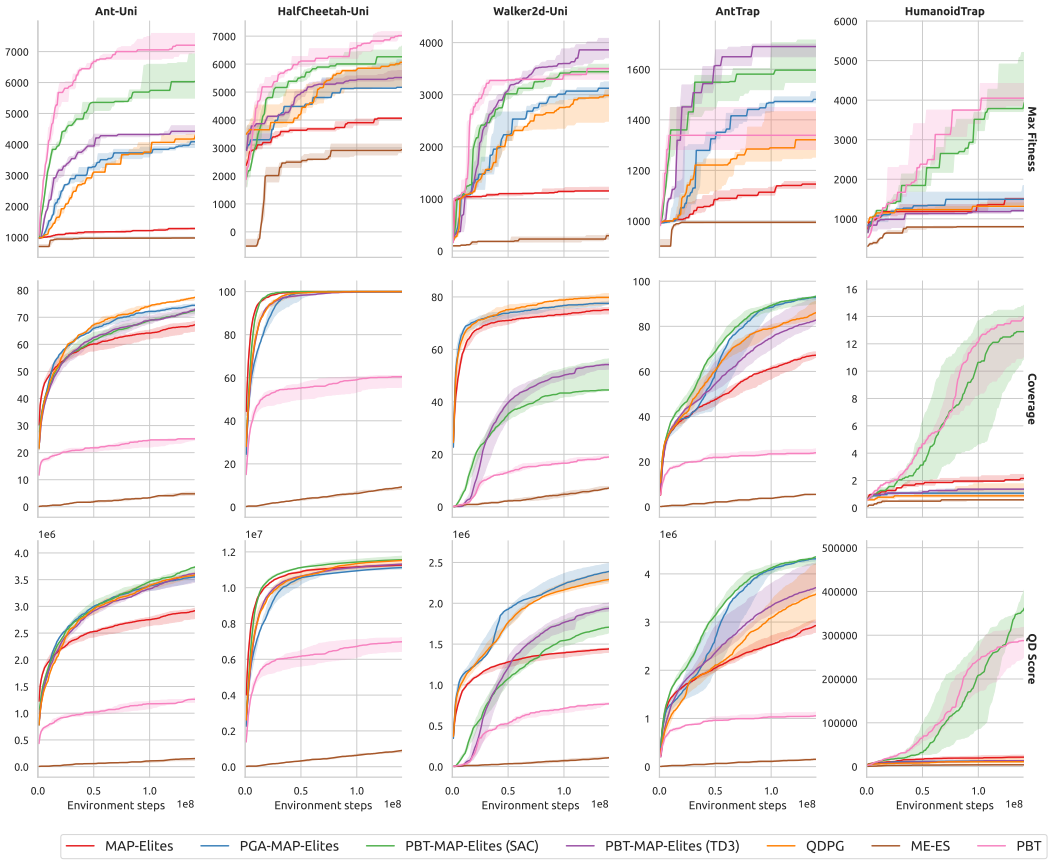

Figure 2: Performance comparison of PBT-MAP-ELITES with baselines on the basis of standard metrics from the QD literature for five environments from the QDAX suite (which is based on the BRAX engine). We benchmark two variants of PBT-MAP-ELITES, one where it is composed with SAC and one where it is composed with TD3. All methods are trained with a total budget of $N = 1.5e8$ environment timesteps. Experiments are repeated over 5 runs with different random seeds and the medians (resp. first and third quartile intervals) are depicted with full lines (resp. shaded areas).

ronments while the reward at each timestep is the velocity of the robot's center of gravity projected onto the forward direction.

2. ANT-TRAP and HUMANOID-TRAP are environments with deceptive reward signals used in the QD-RL literature to evaluate an algorithm's ability to solve complex continuous control problems that require a good amount of exploration, see Colas et al. (2020); Conti et al. (2018); Pierrot et al. (2022). In these environments, the goal is also to make the legged robot run as fast as possible in the forward direction, though with the additional difficulty that the robot is initially facing a trap. As a result, following the reward signal in a greedy fashion leads the robot into the trap. The robot must explore the environment and learn to go around the trap, even though it is temporarily suboptimal, in order to obtain higher returns. In these environments, the behavior descriptor is defined as the position of the robot's center of gravity at the end of an episode. All of these environments are based on the BRAX simulator Freeman et al. (2021) and are available in the QDAX suite Lim et al. (2022).

Setup. We compare PBT-MAP-ELITES to state-of-the-art MAP-ELITES-based methods, namely MAP-ELITES, ME-ES, PGA-MAP-ELITES as well as QD-PG. For these experiments, we benchmark two variants of PBT-MAP-ELITES: one where it is composed with SAC and one where it is composed with TD3. For the sake of fairness, we use the same values for parameters that are used by multiple methods. In particular, all MAP-ELITES-based methods maintain a repertoire of 1024 cells and use CVT with the same parametrization to discretize the behavior descriptor space into 1024 cells. Similarly, when a variation operator is needed, we always use the isoline operator with the same pa-

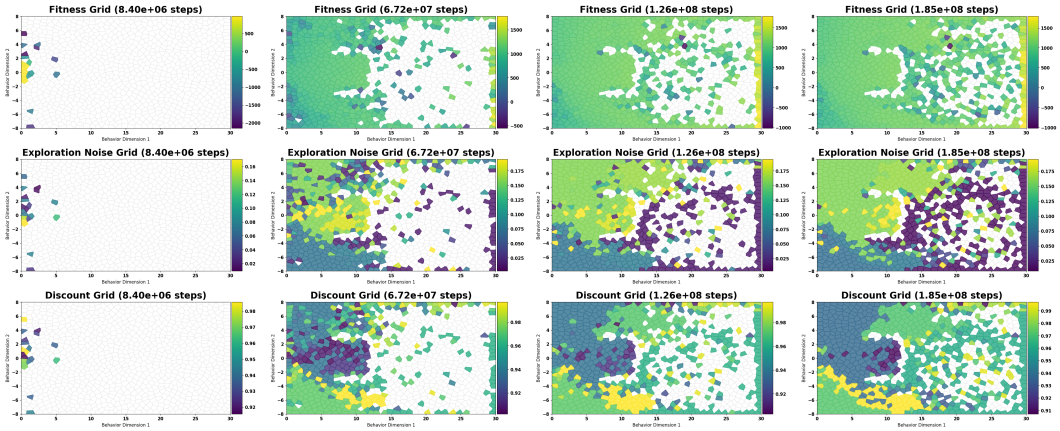

Figure 3: Visualizations of the performance and hyperparameters of the agents stored in the MAP-ELITES repertoire of PBT-MAP-ELITES (TD3) at different time points during training on the ANT-TRAP environment. The first row corresponds to the fitnesses of the agents. The second row corresponds to the exploration noise added by the agents to the actions returned by their policies when collecting experience in exploration mode. The third row corresponds to the discount factor γ used by the agents. Snapshots of the repertoire are taken at uniformly spaced intervals during training.

rameters $\sigma_1 = 0.005$ and $\sigma_2 = 0.05$. All policy and critic networks are implemented by two MLPs layers with 256 hidden neurons per layer. For methods relying on the TD3 agent, the hyperparameters used are the ones introduced in the original paper for MuJoCo environments. Pseudocodes and parameter values for all algorithms under study are provided in the Appendix.

Additionally, we compare PBT-MAP-ELITES to the PBT algorithm Jaderberg et al. (2017) (pseudocode provided in the Appendix) when it is used to optimize populations of SAC agents. Both PBT-MAP-ELITES and PBT evolve populations of 80 agents and use the same ranges for the hyperparameters. All policy and critic networks are implemented by two-layer MLPs with 256 hidden neurons per layer, just like for TD3 for PGA-MAP-ELITES and QD-PG. Furthermore, the parameters of all agents in the population are identically initialized. For PBT-MAP-ELITES (resp. PBT), agents in the bottom $p = 0.2$ (resp. $p = 0.4$) fraction of the population (in terms of fitness) are replaced by agents sampled from the top $n = 0.1$ fraction of the population. For PBT-MAP-ELITES, a fraction of $k = 0.4$ of the agents that are neither in the bottom 20% nor in the top 10% of the population are replaced by agents randomly sampled from the MAP-ELITES repertoire. All other parameters and design choices are identical for these two methods.

Metrics and fair comparisons. Following standard practice in the QD literature, we monitor three metrics used to evaluate the performance of a collection of policies during training. 1. We measure the **maximum fitness**, defined as the maximum expected return across policies in the collection. 2. We measure the **coverage** over the descriptor space, computed as the number of cells that have been filled. 3. We measure the **QD-score**, computed as the sum of fitnesses attained by the policies stored in the repertoire. For this last metric to be meaningful, we assume that fitnesses are all non-negative. If not, a positive value is added to all fitnesses to enforce it. In any case, this value is the same for all methods for fairness. Since some of the metrics require a repertoire to be properly defined, we introduce a passive repertoire for PBT to be able to evaluate it on the same basis as the other methods. Specifically, at the end of each PBT iteration, the population of agents generated by PBT is evaluated and inserted into a repertoire. For each method, we report the evolution of these metrics w.r.t. the total number of interactions with the environment. Note that while the evaluation of an agent contributes to the total number of interactions for MAP-ELITES-based methods, this is not the case for PBT as the evaluations are only used to estimate the metrics for this method.

6 Results and Discussion

Statistics on QD metrics are reported for all environments and methods on Figure 2.

Performance comparison to other MAP-ELITES-based methods. We observe that PBT-MAP-ELITES (SAC) is the only method able to solve HUMANOID-TRAP within the allocated timestep budget, outperforming all the other methods by a significant margin. HUMANOID-TRAP is a challenging environment as getting high returns requires not only to get the humanoid robot to run, which is a challenging continuous problem in itself, but also to learn to sidestep the trap in spite of a deceptive reward signal. This environment, first introduced in Colas et al. (2018), has remained out of reach for MAP-ELITES-based methods, setting aside ME-ES which solves it with a timestep budget two orders of magnitude higher. Interestingly, the maximum fitness remains below 2000 for methods based on the TD3 agent, which means they were not able to get the humanoid robot to run at all. This is a testament to the difficulty of the problem. Recall that TD3 was not able to solve the MuJoCo-based version of the Humanoid environment in the original paper that introduced this algorithm Fujimoto et al. (2018). A careful tuning of the algorithm design choices and hyperparameters, carried out in a later study, was required to get TD3 to perform well on this environment. Setting aside the WALKER2D-UNI environment, note that PBT-MAP-ELITES (SAC) either outperforms, often by a significant margin for the maximum fitness metric, or performs on par with MAP-ELITES-based methods. Interestingly, the SAC variant of PBT-MAP-ELITES often performs better than the TD3 variant, but not always. On a side note, we also observe that ME-ES surprisingly gets outperformed by all MAP-ELITES competitors, including the original MAP-ELITES algorithm, in all environments. This can be explained by the fact that ME-ES uses 1000 evaluations (i.e. $1e6$ timesteps) to update a single policy. As a result, for a repertoire consisted of 1024 cells and with a budget of $1.5e8$ timesteps, the maximum coverage that can be reached by ME-ES is 15% only. In the original study, ME-ES manages to outperform other MAP-ELITES-based methods with a budget of $1e10$ timesteps.

Performance comparison to PBT. We observe that PBT outperforms the SAC variant of PBT-MAP-ELITES in terms of maximum fitness on HALFCHEETAH-UNI and ANT-UNI. This is expected as: (1) these environments do not require a significant amount of exploration, (2) PBT only aims to maximize the maximum fitness, and (3) PBT-MAP-ELITES aims to maximize both the maximum fitness and the policies' diversity. However, we observe the opposite trend on ANT-TRAP and HUMANOID-TRAP where significant exploration is required to achieve high returns given the deceptive nature of the reward signal. We conclude that optimizing for diversity turns out to play a crucial role for these two environments. As expected, PBT-MAP-ELITES outperforms PBT in terms of coverage and QD-score in all environments, setting aside HUMANOID-TRAP. The seemingly unexpected results observed on HUMANOID-TRAP stem from the fact that covering the behavior descriptor directly correlates with exploration of the (x, y) space, which is required to achieve high returns in this environment due to the presence of the trap.

Repertoire interpretation. By visualizing the evolution of the fitnesses and hyperparameters of the agents stored in PBT-MAP-ELITES's repertoire at different time points during training, see Figure 3, we observe that PBT-MAP-ELITES evolves locally-coherent (w.r.t. the descriptor space) maps of hyperparameters that change significantly during training. In particular, we remark that PBT-MAP-ELITES dynamically increases the amount of exploration noise of the TD3 agents to boost exploration when needed to go around the trap and decreases this parameter once the trap has been sidestepped to focus on getting high returns. This mechanism gives a significant advantage to PBT-MAP-ELITES over QD-PG and PGA-MAP-ELITES, for which this parameter is set to a constant value.

7 Conclusion

In this work, we revisit the standard formulation of the QD neuro-evolution problem by evolving repertoires of full agents (including hyperparameters among other things) as opposed to only policies. This extension brings flexibility compared to existing frameworks as it allows us to combine any RL algorithm with MAP-ELITES in a generic and scalalable fashion. This formulation also allows us to dynamically learn the hyperparameters of the underlying RL agent as part of the regular training process, which removes a significant burden from the user. Surprisingly, we observe that learning the hyperparameters improves both the asymptotic performance and the sample efficiency in practice for most of the environments considered in this work. Our method is the first to solve the HUMANOID-TRAP environment with less than one billion interactions with the simulator, to be compared with tens of billions of interactions for state-of-the-art QD methods. We hope that this work constitutes one more step towards bridging the gap between Neuro-Evolution and Reinforcement Learning, combining the best of both worlds in a simple framework.

REFERENCES

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. GEP-PG: decoupling exploration and exploitation in deep reinforcement learning algorithms. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1038–1047. PMLR, 2018. URL http://proceedings.mlr.press/v80/colas18a.html.

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

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

Antoine Cully and Yiannis Demiris. Hierarchical behavioral repertoires with unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 69–76, 2018.

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

Arthur Flajolet, Claire Bizon Monroc, Karim Beguir, and Thomas Pierrot. Fast population-based reinforcement learning on a single machine. In *International Conference on Machine Learning*, pp. 6533–6547. PMLR, 2022.

Matthew C. Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 10040–10052, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/532923f11ac97d3e7cb0130315b067dc-Abstract.html.

Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pp. 94–102, 2020.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax – a differentiable physics engine for large scale rigid body simulation, 2021. URL https://arxiv.org/abs/2106.13281.

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

Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary computation*, 26(3):381–410, 2018.

Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Are quality diversity algorithms better at generating stepping stones than objective-based search? In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 115–116, 2019.

Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Procedural content generation through quality diversity. In *2019 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2019.

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.

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.

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

Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Beslon, David M Bryson, et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *Artificial life*, 26(2):274–306, 2020.

Bryan Lim, Maxime Allard, Luca Grillotti, and Antoine Cully. Accelerated quality-diversity for robotics through massive parallelism. *arXiv preprint arXiv:2202.01258*, 2022.

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

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

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

Thomas Pierrot, Valentin Macé, Félix Chalumeau, Arthur Flajolet, Geoffrey Cideron, Karim Beguir, Antoine Cully, Olivier Sigaud, and Nicolas Perrin-Gilbert. Diversity policy gradient for sample efficient quality-diversity optimization. In Jonathan E. Fieldsend and Markus Wagner (eds.), *GECCO '22: Genetic and Evolutionary Computation Conference, Boston, Massachusetts, USA, July 9 - 13, 2022*, pp. 1075–1083. ACM, 2022. doi: 10.1145/3512290.3528845. URL https://doi.org/10.1145/3512290.3528845.

Aloïs Pourchot and Olivier Sigaud. CEM-RL: combining evolutionary and gradient-based methods for policy search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BkeU5j0ctQ.

Nemanja Rakicevic, Antoine Cully, and Petar Kormushev. Policy manifold search: exploring the manifold hypothesis for diversity-based neuroevolution. In Francisco Chicano and Krzysztof Krawiec (eds.), *GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021*, pp. 901–909. ACM, 2021. doi: 10.1145/3449639.3459320. URL https://doi.org/10.1145/3449639.3459320.

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.

Anurag Sarkar and Seth Cooper. Generating and blending game levels via quality-diversity in the latent space of a variational autoencoder. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, pp. 1–11, 2021.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Yujin Tang, Yingtao Tian, and David Ha. Evojax: Hardware-accelerated neuroevolution. *arXiv preprint arXiv:2202.05008*, 2022.

Bryon Tjanaka, Matthew C. Fontaine, Julian Togelius, and Stefanos Nikolaidis. Approximating gradients for differentiable quality diversity in reinforcement learning. In Jonathan E. Fieldsend and Markus Wagner (eds.), *GECCO '22: Genetic and Evolutionary Computation Conference, Boston, Massachusetts, USA, July 9 - 13, 2022*, pp. 1102–1111. ACM, 2022. doi: 10.1145/3512290.3528705. URL https://doi.org/10.1145/3512290.3528705.

Vassiiis Vassiliades and Jean-Baptiste Mouret. Discovering the elite hypervolume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 149–156, 2018.

Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation*, 22(4):623–630, 2017.

Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 4015–4023. PMLR, 2021.