Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Combining MAP-Elites with Reinforcement Learning

Author:
Mathilde Outters

Supervisor:
Antoine Cully

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

September 2022

Abstract

Taking inspiration from natural evolution and genetics, Evolutionary Algorithms are a family of powerful methods to optimise black-box functions. Solutions represent individuals in a population, which evolves thanks to selection and local exploration mechanisms. Quality-Diversity [1] (QD) emerged as a promising sub-field, adding novelty search to this setting. Instead of taking the direct path to improve the fitness, QD algorithms seek to improve not only the performance but also the novelty of each solution. Behavioural descriptors are used to define different types of solutions and novelty is achieved by maximising the distance between these descriptors. Ultimately, QD algorithms aim to find diverse ways to solve a task, by collecting locally-optimised solutions spread across the behavioural space. This exploration ability helps reaching stepping stones and makes QD algorithms of particular interest in sparse and deceptive-reward environments. Such divergent search has successfully been used in the MAP-Elites [2] algorithm to learn efficient and diversified walking gaits for robots.

The more recent Policy Gradient Assisted MAP-Elites (PGA MAP-Elites) [3] incorporates directed performance improvement tools from Deep Reinforcement Learning. The addition of an Actor-Critic scheme efficiently drives the exploration of solutions towards more promising regions of the solution space by using the Critic's gradient estimates in gradient-based variations. This data-efficiency allows PGA-MAP-Elites to scale to the optimisation of policies with a large number of parameters, such as deep neural network (DNN) controllers. This approach is however limited by the unaligned goals of Quality-Diversity and Reinforcement Learning. The latter's single-agent setting aims to learn the single best policy to solve a task. In PGA MAP-Elites, the Critic will evolve solutions towards this global optimum regardless of their specific behaviour, thereby making them converge to a similar behaviour and hindering the optimisation of controllers behaving differently.

To further enhance the promising hybridisation of PGA MAP-Elites, this work proposes to train a "behaviour-aware" Critic that would be capable of advising good time-step actions for a sought final behaviour. To train such a Critic, we use many Actors and let each of them become an expert for a different area of the behaviour space. The results show that this new algorithm significantly improves both the diversity and the performance of the generated repertoire of solutions.

Acknowledgments

I would like to thank Antoine Cully for his supervision and valuable guidance throughout this project.

I would also like to thank Manon Flageat for her friendly support and for always taking the time to answer my questions.

Contents

Chapter 1

Introduction

1.1 Problem Statement

Reinforcement Learning [4] is concerned with a computational agent learning goal-oriented behaviours from interaction and feedback in its environment. At each discrete time step, the agents performs an action a, which transitions the agent into a new state s, and thereby receives a reward r. The Markov Decision Process (MDP) framework formulates this sequential decision-making process as a discrete-time stochastic control process $(S, A, P^a_{ss'}, R^a_{ss'})$ with given probabilities of transitioning from a current state into a new state by executing a given action $p(s_{t+1}|s_t, a_t)$ and thereby receiving a reward $r_t \sim p(r_{t+1}|s_t, a_t)$.

Learning which action to take given a state, or best *policy*, so as to maximise the cumulative sum of a scalar reward signal can be formulated as a black-box score function optimisation problem. Deep Reinforcement Learning (DRL) [5] incorporates deep learning into this framework. The function approximation power of deep neural networks is used to represent the learned policy that can take the raw high-dimensional state of the agent as input and output the best action to take in that state, with respect to a predefined goal-related objective function. By discarding the need to manually engineer the state space, DRL allows robotic agents to learn to make decisions by trial and error from unstructured data like an image or raw sensor data. For a robot agent, learning to walk in a given environment fits into this sequential decision-making framework, where the available actions are the sets of torques applied to its joints, the reward could be the distance travelled after a fixed period of time, and the states could be the positions of each of the joints at every time-step. This scaling of Reinforcement Learning to complex high-dimensional environments has demonstrated many interesting application in robotics [6].

As stated above, Deep Reinforcement Learning's objective is to find the optimal policy to solve a given task in the environment. In the robotics field however, they are cases where such objective would not be the most relevant. Let's think of the case of post-damage recovery, where a robot would need to quickly adapt to a damage [7]. In practice, this need for adaptation translates in having an available collection

of policies to chose from. To avoid the need to foresee all possible failure modes, this collection should cover a broad range of possible behaviours with the best local solutions. This is precisely the aim of Quality-Diversity (QD) algorithms [1], which seek to "illuminate" this behaviour space, explicitly maintaining a large repertoire of best solutions for every region of the behavioural space.

The reason why one might have interest to look for a large set of solutions instead of a single optimal one is twofold. First, for robustness. As already mentioned, a repertoire of behaviours generated before the robot is deployed can be exploited for adaptation by selecting the most appropriate behaviour in a given situation once the robot is active in the environment. Second, in domains where the fitness function can be misleading, such as a deceptive maze, the search for diversity drives the algorithm to explore more of the feature space. This exploration might ultimately lead to find different, and often better, fitness peaks whereas purely performance-driven optimisation method would get stuck in local optima.

1.2 Objectives

The purpose of this project is to investigate ways of generating highly-performing behavioural repertoires that use deep neural network controllers for locomotion tasks. PGA MAP-Elites [3] (see Section 2.3.3), proposed the fruitful idea of using a gradient-based variation operator to perform QD optimisation. Their Policy Gradient (PG) operator inspired by DRL is used to evolve solutions with gradient descent, using fitness gradients derived by a critic neural network. The overall objective of this project is to investigate further improvements of the algorithm.

In particular, we would like to build on the proposition made in [8] to use a "behaviour-aware" Critic in the Policy Gradient variations. This is motivated by the fact that solutions optimised with the PG variation operator will tend to evolve towards a behaviour similar to the greedy actor, which is contrary to QD objectives. Hence we would like to add more expressiveness to the Critic and make it generalise not only over states and actions but also over behavioural goals. We investigate ways of allowing the critic to focus on many different regions of the behavioural space at the same time and learn a value function which is conditioned on an intended behaviour. For a given solution in the archive, this new Critic should suggest, for each sampled state, a relevant action given a sought final behavioural descriptor (BD).

1.3 Outcomes

Our results demonstrate that we were able to successfully train a BD-aware Critic. Additionally, our results show that using this new Critic in the PG variations allows to increase the performance of the algorithm. The generated repertoires show more spread-out good solutions within much fewer iterations of the algorithm.

Chapter 2

Background and related work

2.1 Quality-Diversity Algorithms

Evolution on Earth has lead to the existence of many species, each one being the visible tip of a long branch of variations. Each branch traces the history of optimising steps, constrained by the given's species environment. Each species is only competing and optimised to survive its local ecological *niche*. Taking inspiration from this observation, Quality-Diversity (QD) algorithms [1] aim to find large collections or *archives* of high-performing and diverse solutions, by looking for different ways (as defined by a diversity score) to best solve a task (as defined by a performance score).

Exploration of the solution space Φ, also called *genotype* space, is done by stochastically producing new solutions from existing ones (mutating their *genome*). The performance of new a solution ϕ is measured by a fitness function $F : \Phi \to \mathbb{R}$ which highly depends on the task, while diversity is measured by the distance between the solutions in some behavioural space $\mathcal{B}$, whose dimensions are built from the diversity sought (usually lower dimensional than the search space Φ). The solution in each niche is optimised by the Darwinian principles of natural selection: when two solutions exist within the same niche, only the fittest survives in the collection. This principle ensures that the final collection is both *qualitative* and *diverse*, i.e. covering the range of possible behaviours.

The general QD framework defines a common loop of steps:

1. Stochastic selection of *parents* in the archive (uniform or objective-biased)
2. Random variations of selected solutions (through *mutations* and *crossovers*)
3. Evaluation of the performance of the new solutions and projection in $\mathcal{B}$
4. Tentative addition in the archive, based on both diversity and performance.

A note on terminology: in this work, the words *solutions* and *controllers* are often used interchangeably, they both refer to genotypes of individuals in the archive ϕ. The genotype is used to control the system and derive a *phenotype*, i.e. the associated

trajectory in the environment, upon which the fitness is evaluated. The behavioural descriptor (BD) $\xi(\phi)$ is the projection of the trajectory in the smaller-dimension behaviour space $\mathcal{B}$, it describes the "type" of each found solution.

2.1.1 MAP-Elites

Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [7; 2] enforces diversity and local competition in the collection by discretizing the behavioural space $\mathcal{B}$ into a grid and searching for the best solutions for every cell of the grid. To mitigate the exponentially increasing number of cells with the number of dimensions of $\mathcal{B}$, Centroidal Voronoi Tessellation MAP-Elites (CVT-MAP-Elites) [9] introduces an automated partition of the behaviour space into an explicitly controlled number k of convex polygonal-shaped bins, whose centroids are maximally spread in $\mathcal{B}$.

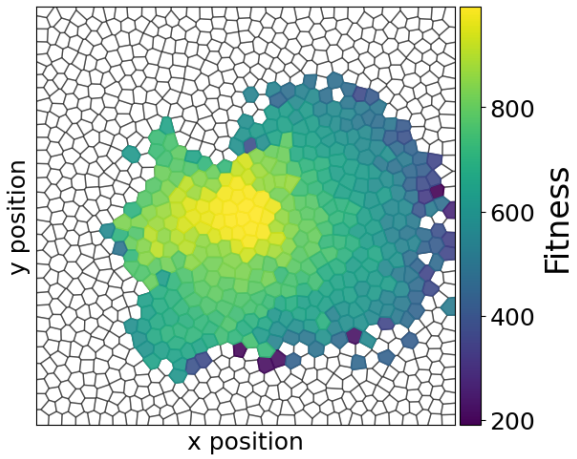


Figure 2.1: Example of a CVT archive after some iterations of MAP-Elites. In this robot omnidirectional locomotion case, the behaviour space $\mathcal{B}$ has 2 dimensions: a solution's behavioural descriptor (BD) is defined as the x and y final position of the robot. Each cell is a "region" of $\mathcal{B}$ that we aim to optimise. Each new individual ϕ is characterised by a BD $\xi(\phi)$ and is evaluated to derive an associated fitness $f(\phi)$. If the cell in which the BD lands is empty, the solution gets added to the archive (*novelty*), otherwise it will compete with the occupying solution and the best performing will survive in the archive (*quality*). This mechanism achieves local (intra-niche) competition of the individuals.

To understand the motivation behind the future introduction of gradient-information, it is important to stress the variation mechanism used in MAP-Elites. The search space Φ is locally explored through variations of the genotype, which happens in two steps: the selection of parents from the archive (usually uniform) and the creation of new solutions as randomly modified copies of their parent. This "random modification" defines how the algorithm locally moves in the search space and will define the future starting points for exploration.

There are two types of variation operation commonly used: mutations and crossovers. Mutations consist in applying random changes to the genotype of the parent con-

Algorithm 1: Simple CVT MAP-Elites. Adapted from [8]

Input : Search space Φ, Behaviour space B, Fitness function $F : \Phi \to \mathbb{R}$, Descriptor extraction function $\xi : \Phi \to B$, Evaluation resource max_eval, Generation size K.
Output: Final Archive $\mathcal{X}$

$\mathcal{X} \leftarrow create\_empty\_cvt\_archive(B)$
$\phi_1, ..., \phi_K \leftarrow random\_solutions(\Phi, K)$
$\mathcal{X} \leftarrow add\_to\_archive(\phi_1, ..., \phi_K, \mathcal{X})$
$nb\_eval \leftarrow 0$
while $nb\_eval < max\_eval$ **do**
  $\phi_1, ..., \phi_K \leftarrow selection(\mathcal{X})$
  for $i \leftarrow 0$ **to** K **do**
    $\hat{\phi}_i \leftarrow variation(\phi_i, \mathcal{X})$
    $f_i, b_i \leftarrow evaluate(\hat{\phi}_i, F, \xi)$
    if $cell\_is\_empty(b_i, \mathcal{X})$ **then**
      $\mathcal{X} \leftarrow add\_to\_archive(\phi_i, \mathcal{X})$
    else
      $\phi_{in\_archive} \leftarrow select(\mathcal{X}, get\_cell(b_i))$
      $f_{in\_archive}, b_{in\_archive} \leftarrow retrieve\_evaluation(\phi_{in\_archive})$
      if $f_i > f_{in\_archive}$ **then**
        $\mathcal{X} \leftarrow add\_to\_archive(\phi_i, \mathcal{X})$
      end if
    end if
  end for
  $nb\_eval \leftarrow nb\_eval + K$
end while

troller. In practice, a stochastic noise is randomly applied and this noise can be drawn from different distributions. Several mutation operators have been proposed in the literature: gaussian, uniform, polynomial [10]. A crossover consists in randomly mixing the genotype of two parent controllers. Two commonly used methods are Simulated Binary Crossover with One Offspring (SBX) and Iso+LineDD, both introduced by [10]. The Iso+LineDD variation operator combines a Gaussian mutation with a noisy crossover between the two parents, where the hyperparameters σ_1 and σ_2 respectively control the Gaussian perturbation and the interpolation noise. The parents ϕ_i and ϕ_j are sampled from the archive and the new individual is generated as:

$$\hat{\phi} = \phi_i + \sigma_1 \mathcal{N}(\mathbf{0}, I) + \sigma_2(\phi_j - \phi_i)\mathcal{N}(\mathbf{0}, 1)$$

All the above Genetic Algorithm (GA) variations lack directed search power, making them data-inefficient. The sole mechanism implicitly orienting the search towards promising regions of Φ is the adding rule: solutions get added to the collection only if they are novel (they fill an empty cell) or perform better (higher fitness) than the current local solution. No information about the gradient of the fitness is used.

2.2 Deep Reinforcement Learning

As already discussed in Section 1, Reinforcement Learning [4] is concerned with an agent acting sequentially over the course of an episode. A Markov Decision Process $(S, A, P^a_{ss'}, R^a_{ss'}, \gamma)$ is used to model the interaction of the agent with its environment. The agent's behaviour is governed by an internal probability distribution or *policy* $\pi(a|s)$. For a desterministic policy, the agent finds itself in a state s_t at each discrete time-steps $t \in [0, T]$, and chooses an action $a_t = \pi(s_t)$ accordingly.

The objective of RL methods is to find the policy that the agent should follow from an initial state s_0 to maximise the expected discounted sum of future rewards, also called *return* $R_0 = \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$, where $\gamma \in [0, 1]$ is a discount factor to account for how much we care about immediate rewards. This expected return formalises the performance of a policy $J(\pi) = \mathbb{E}_{s_0}[R_0|s_0]$.

Deep Reinforcement Learning's use of deep learning brings important benefits to this framework. The function-approximation capabilities of DNNs to model policies π_ϕ, parameterised by the network's weights ϕ, allows to scale RL to high-dimensional problems. Training the policy network can be done much more efficiently thanks to the powerful directed search abilities of neural networks' gradient-based learning.

2.2.1 Deep Q-Networks (DQN)

The state-action value function $Q^\pi(s_t, a_t)$ of a policy π quantifies the interest the agent should have of taking an action a_t in s_t and subsequently follow π. To find the optimal policy π^*, RL methods seek to find the optimal Q-function $Q^* = Q^{\pi^*}$ which satisfies a self-consistent constraint known as the Bellman Optimality equation:

$$Q^*(s, a) = \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q^* (s', a') \right]$$

Once the state-action function Q converges towards Q^*, the optimal policy is found by simply acting greedily w.r.t Q^*, i.e. $\pi^*(s_t) = \text{argmax}_{a \in \mathcal{A}} Q(s, a)$.

Deep Q-Networks (DQN) [11] is a model-free RL algorithm, using a neural network Q_θ trained with gradient descent to approximate Q^*. Model-free algorithms do not rely on knowing the dynamics of the MDP and solely learn from interaction with the environment. The training phases are intertwined with experience phases. During the latter, an exploratory policy based on the current approximation of Q is used to generate episodes, and the collected transition tuples (s_t, a_t, r_t, s_{t+1}) are stored in the *replay buffer*. At each gradient descent step of the network training, a mini-batch of transitions are sampled from this replay buffer and the Q-network is trained using the loss $\mathcal{L} = \mathbb{E}[(Q(s_t, a_t) - y_t)^2]$, where $y_t = r_t + \gamma \max_{a' \in \mathcal{A}} Q(s', a')$, which encourages the Q-function to satisfy the Bellman equation.

2.2.2 Policy Gradients

Value-based methods such as described above, rely on learning a value function to assess the quality of the decisions and deriving the policy by selecting the actions with best estimated values. This approach becomes intractable for high-dimensional state and action spaces. Instead, policy-based methods explicitly search for the optimal policy by directly optimising the policy parameters, i.e. the parameterised policy π_ϕ can select actions without consulting a value function.

Policy gradient (PG) methods rely upon using the gradient of the expected return $\mathbb{E}_{\pi_\phi}[r(\tau)]$, in order to optimise the policy weights with gradient ascent steps $\phi \leftarrow \phi + \alpha \nabla_\phi J(\phi)$. The gradient of this expected return is computed using the Policy Gradient Theorem [12], where $\tau = \{(s_t, a_t)\}_{t \in [0,T]}$:

$$\nabla_\phi J(\phi) = \mathbb{E}_{\tau \sim \pi_\phi(\tau)}[r(\tau) \nabla_\phi \log \pi_\phi(\tau)]$$

Expanding the $\log \pi_\phi(\tau)$ term, it can be shown that the analytical form of the performance gradient becomes independent of the environment's transition dynamics:

$$\nabla_\phi J(\phi) = E_{\tau \sim \pi_\phi(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\phi \log \pi_\phi \left(\mathbf{a}_t \mid \mathbf{s}_t \right) \right) \left(\sum_{t=1}^{T} r \left(\mathbf{s}_t, \mathbf{a}_t \right) \right) \right]$$

In practice, the expectation term is estimated by sampling a mini-batch of traces and averaging them out, as used in the REINFORCE algorithm [13]:

$$\nabla_\phi J(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\phi \log \pi_\phi \left(\mathbf{a}_{i,t} \mid \mathbf{s}_{i,t} \right) \right) \left(\sum_{t=1}^{T} r \left(\mathbf{s}_{i,t}, \mathbf{a}_{i,t} \right) \right)$$

Although conceptually powerful and working with continuous action spaces, the above method is unstable due to the high variance in the sampled trajectories.

2.2.3 Actor-Critic Methods

The Actor-Critic scheme splits the task of learning the best policy π^* between two agents training collaboratively. They perform their task autonomously but use feedback from each other to improve. The first agent is called the *actor* (or target policy π) and is in charge of collecting experience data by acting in the environment. The second agent is called the *critic* and approximate the actor's action-value function Q_π. During training, the actor aims to select actions that are evaluated higher by the critic and the critic aims to give more accurate evaluations to the actions selected by the actor. The parameterised value function Q_π helps to reduce the variance of the policy gradient estimates.

In the Deep Deterministic Policy Gradients (DDPG) [14] algorithm, the Actor network $\pi_\phi : \mathcal{S} \rightarrow \mathcal{A}$ outputs an action based on a state (policy model) while the Critic network $Q_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ approximates the action-value function (value model). The empirical return $r(\tau)$ in the policy gradient updates is replaced by a bootstrap

estimate of the next state provided by the Critic, reducing the high variance in the sampled trajectories and allowing to learn at each time step. Thus, the parameterised policy (Actor) can still select actions without consulting a value function but the values of actions learned by the Critic are used to update the policy weights.

Actor-critic methods in continuous control domains can suffer from instability and are prone to find sub-optimal policies because of the function approximation error in both the actor's policy and the critic's value that reinforce each other. To improve the stability of the training process, the Deep Deterministic Policy Gradients (DDPG) algorithm [14] also incorporated the replay buffer $\mathcal{B}$ and the target network $Q_{\theta'}$ introduced by [11].

The Q-value function learned off-policy by the critic can in theory be used to output the best deterministic policy by taking the best action in each state: $\pi^*(s) = a^* = \text{argmax}_a\, Q^*(s, a)$. However, for continuous action space this maximisation operation would require too many evaluations and becomes computationally infeasible. Instead, we learn a deterministic target policy $\pi_\phi(s)$ (approximating the best action) but keeping the actor's policy stochastic by adding exploration noise when selecting the actions. This is a form of off-policy learning, which enables the use of transitions from another policy to improve the current policy.
The actor π_ϕ and critic Q_θ are trained together. At each time step, a minibatch of transitions is sampled, the critic is updated thanks to mean-squared Bellman error and the actor thanks to the deterministic policy gradient theorem [15], using an empirical average over the minibatch to approximate the expectation term:

$$\nabla_\phi J(\phi) = \mathbb{E}_{s\sim\rho^\pi}[\nabla_\phi \pi_\phi(s) \nabla_a Q^\pi(s, a) \mid_{a=\pi_\phi(s)}]$$

2.2.4 TD3

The Twin Delayed DDPG (TD3) [16] builds on DDPG and further improve the stability of the actor-critic setup by adding three new elements:

- *Clipped double Q-learning*: taking the minimum value between a pair of independently trained target critics stabilises the learning target by limiting overestimation. The new critic target update $y = r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi'}(s'))$ enforces states with lower variance estimation error to have higher values.

- *Delayed policy updates* allow to minimize the variance and error in the Q-network's predictions before updating the policy network.

- *Target policy smoothing regularisation*: When updating the critic, simply taking the target action $\pi_{\phi'}(s')$ given by a deterministic actor would have high variance due to function approximation inaccuracies. Instead, the idea is to smooth this learning target over $Q_{\theta'}$ so that the critic fits to a small area around the target action: $y = r + \mathbb{E}_\epsilon[Q_{\theta'}(s', \pi_{\phi'}(s') + \epsilon)]$, where $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$. The learned value function is hence "safer" as it provides higher value to actions resistant to

Algorithm 2: TD3 algorithm. Adapted from [16]

Input : Initialised critics networks $Q_{\theta 1}$, $Q_{\theta 2}$ and actor network π_ϕ with random parameters θ_1, θ_2, ϕ.
Hyperparameters for policy update delay d, exploration σ, robustness noise $\tilde{\sigma}$,
clipping for smooth target policy c, target nets slow-moving update rate τ.
Output: Trained actor π_ϕ and critic Q_θ.
Initialise target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$.
Initialise replay buffer B.
for $t \leftarrow 1$ **to** *n_iter* **do**
  One-step interaction of the actor with the environment:
  // Select action with exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$
  Select $a \sim \pi_\phi(s) + \epsilon$
  Store transition tuple (s, a, r, s') in $\mathcal{B}$.
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  // Sample policy smoothing noise for robustness
  $\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  // Predict next action with target actor and add noise
  $\tilde{a}_n \sim \pi_{\phi'}(s'_n) + \epsilon_n$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$ // Clipped double Q-learning target
  $\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum_n (y_n - Q_{\theta i}(s_n, a_n))^2$ // Update critics
  if $t \mod d$ **then**
    // Delayed policy update
    Use gradient ascent to update ϕ with estimated deterministic policy gradient:
    $\nabla_\phi J(\phi) = \frac{1}{N} \sum_n \nabla_a Q_{\theta 1}(s_n, a_n) \mid_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s_n)$
    $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$ // Update target critics, $i = 1, 2$
    $\phi' \leftarrow \tau\phi_i + (1-\tau)\phi'_i$ // Update target actor network
  end if
end for

perturbations, which can be useful in stochastic domains. In practice, a clipped random noise is added to the target policy and the TD error is averaged over a mini-batch of of nearby actions estimates.

2.3 Mixing QD and RL

From the above sections, we can see complementary advantages and drawbacks rising from DRL and QD methods. On the one hand, using divergent search for random diversity is sample-inefficient and does not scale to high-dimensional problems. In addition, the found solutions can be unstable as the optimisation process does not account for the evaluation of nearby solutions (no gradient). On the other hand,

PG approaches perform directed search by exploiting the analytical structure of the objective function but suffer from insufficient exploration mechanisms. In hard problems with a sparse or deceptive reward signal this leads the gradient to sub-optimal solutions.
Recent works, presented bellow, have successfully combined PG-driven and GA-driven variations. The RL framework is blended into QD methods, by taking the fitness of a controller $F(\phi_i) = f_i$, as the equivalent of the corresponding policy's expected return $J(\phi_i)$. The general motivation is to to scale MAP-Elites to high-dimensional controllers parameterised by large neural networks.

2.3.1 MAP-Elites-ES

MAP-Elites with Evolution Strategies (ME-ES) [17] leverage the optimisation performance of Evolution Strategies (ES) to train DNN controllers. ES are gradient-based optimisation algorithms that gradually move a population of solution towards promising regions of the search space. The underlying idea is to use neighbour solutions to approximate the objective gradient in the region of the parent solution being evolved. Natural ES represent each offspring population as a parameterised distribution $p_\psi(\phi)$. At each generation, the individuals of the offspring population of size N are sampled as $\phi_i \sim \mathcal{N}(\phi_{parent}, \sigma^2 I)$ with fixed variance σ^2. Using the gradient approximation from REINFORCE [13]:

$$\nabla_\phi J(\phi) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\log \pi_\phi \left(\tau_i\right) r \left(\tau_i\right)}_{\sum_{t=1}^{T} \nabla_\theta \log_\phi \pi_\phi(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}$$

They optimise $\mathbb{E}_{\theta \sim p_\psi} F(\theta)$ with the fitness gradient approximation:

$$\nabla_\psi \mathbb{E}_{\phi \sim p_\psi}[F(\theta)] \approx \frac{1}{N} \sum_{i=1}^{N} F\left(\phi_i\right) \nabla_\psi \log_\psi(\phi)$$

In practice, since $\phi_i = \phi + \sigma\epsilon_i$, they use:

$$\nabla_\psi \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[F(\phi + \sigma\epsilon)] \approx \frac{1}{N\sigma} \sum_{i=1}^{N} F\left(\phi_i\right) \epsilon_i$$

The gradient is thus evaluated by using an average displacement of perturbed versions of the parent, weighted by their objective score. This objective can be set to be *exploitation* (as above), with fitness as the objective function, or *exploration*, with novelty as the objective function. Novelty is computed as the average distance between the controller's behavioural descriptor and its k nearest neighbors in the archive containing the BDs of all previously encountered solutions.

Because the number of evaluations in the environment at each generation is much greater than in the original MAP-Elites ($N = 10^4$ instead of 1), the number of generations is reduced. This highlights the crucial role of the starting point in the search

space, i.e. the parent solution from which to initiate the generations. Moving away from uniform sampling used in MAP-Elites, ME-ES uses biased sampling, with two possible strategies: the sampling can be biased for *exploitation*, by selecting from solutions with highest fitness, or for *exploitation*, by selecting from cells with highest novelty scores. To summarize, once a parent controller ϕ has been objective-biased selected in the grid, it is evolved for a fixed number of generations using ES, where each generation aims to investigate around the current controller and find the best empirical direction (with respect to the predefined objective: exploration of exploitation) to take a small step into.

2.3.2 QD-PG

QD-PG [18] leverage RL's data-efficiency to replaces MAP-Elite's genetic mutation operator by two gradient-based update operators, independently exploiting the structure of the two decoupled objective functions of QD: quality and diversity. The classic critic from actor-critic is known as Quality Critic and drives PG-based mutations towards high performance while the Diversity Critic drives PG-based mutations towards new behaviours, explicitly encouraging for exploration.

As policies are characterised by their visited states, the diversity of solutions can be described both at the trajectory or episode level (BD) and at the state level. To quantify this diversity at the time step level, they introduce the state descriptor (SD) space D, and the state descriptor extraction function $\psi : S \to D$.
To estimate the diversity gradient, they use the intuitive idea that the diversity of a population is increased by increasing the *novelty* of a solution ϕ_1 with respect to its J nearest neighbours. This novelty, or summed distance of the solution with respect to its nearest neighbours, can be computed in in behavioural space $\mathcal{B}$, or in the state descriptor space $\mathcal{D}$.
"*The Diversity PG updates the solutions so as to encourage them to visit states with novel SD*" [18], where the novelty of a state descriptor $\psi(s_t)$ is given by time step diversity reward, considering the J SD nearest neighbours of $\psi(s_t)$ in the SD archive A:

$$r_t^D = novelty(s_t, (\phi_j)_{2 \leq j \leq J} = \sum_{j=1}^{J} \|\psi(s_t), \psi(s_j)\|_{\mathcal{D}}$$

The DQ-PG algorithm, although demonstrating high final performance and sample efficiency, comes with some limitations. First, designing the SD extractor ψ such that state diversity correlates with global BD diversity is not a straightforward task. Secondly, every time a PG is computed the corresponding critic (Q/D) is updated. Each evolved solution is an actor for the critic updating it and this training of the critics with various agents can destabilise the process. This requires QD-PG to be used on grids with a lower number of cells, hence limiting the diversity of the final collection.

2.3.3 PGA-MAP-Elites

Policy Gradient Assisted MAP-Elites [3] pairs 2 independent variation operators. The traditional GA variations from MAP-Elites drive divergent search while the gradient-based variation operator drives directed performance improvement. The PG operator relies on the TD3 algorithm presented in subsection 2.2.4. At each generation, the parents are still sampled uniformly from the grid and a fixed proportion n_evo of them is evolved with GA, while the remaining part is evolved with PG.

For a given selected parent ϕ_i, the idea underpinning the PG variation operator is to update ϕ_i such that the actions selected by its policy maximise the critic Q_{θ_1}'s value predictions. In other words, the PG operator updates the given controller towards the approximated Q^*-value gradient.

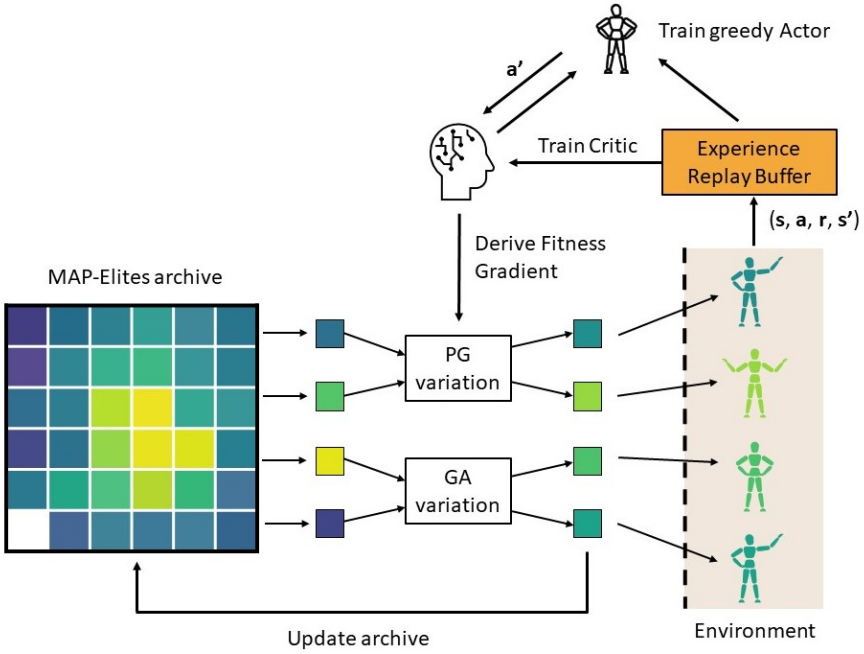


Figure 2.2: Conceptual approach behind PGA-MAP-Elites. A standard MAP-Elites loop is executed repeatedly for selection, variation, evaluation and addition of new solutions to the archive. Variations are equally split between two independent operators: a genetic (GA) variation for divergent search, and a Policy Gradient (PG) variation for directed performance improvement. A Critic neural network is trained in parallel, using experience from the evaluations. Adapted from [3].

To approximate the action-value function, the critic is trained asynchronously to the MAP-Elites loop using the experience collected from all candidate solutions during their evaluations, stored in a replay buffer. The actor ϕ_c, also called *greedy controller*, is a controller of the same architecture as those evolved in the grid, trained together with the critic and used in the critic update target y to predict the next action with maximum action-value. The critic update target comes from TD3: ([16]):

$$y = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta'_i}(s_{t+1}, \pi_{\phi'_c}(s_{t+1}) + \epsilon)$$

Algorithm 3: PGA-MAP-Elites PG Variation Operator. Adapted from [3]

Input : Controller to be evolved π_ϕ,
Critic Q_{θ_1},
Replay memory buffer $\mathcal{B}$,
Number of gradient update steps n_grad,
Learning rate for gradient variation α
Output: New controller $\pi_{\widehat{\phi}}$

$\widehat{\phi} \leftarrow \phi$
// Apply a predefined number of performance gradient ascent update steps to $\pi_{\widehat{\phi}}$
for $i \leftarrow 1$ **to** *n_grad* **do**
  Sample N transitions (s, a, r, s') from $\mathcal{B}$
  // performance gradient estimate
  $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_\phi \pi_\phi(s_t) \nabla_a Q_{\theta_1}(s_t, a)\big|_{a=\pi_\phi(s_t)}$
  $\widehat{\phi} \leftarrow \widehat{\phi} + \alpha \nabla_\phi J(\phi)$ // Usually Adam optimizer is used
end for

Algorithm 4: PGA-MAP-Elites Critic Training. Adapted from [3]

Input : Critic networks Q_{θ_1}, Q_{θ_2},
Actor (*greedy controller*) network π_{ϕ_c},
Target networks $Q_{\theta'_1}$, $Q_{\theta'_1}$, $\pi_{\phi'_c}$,
number of critic update steps n_crit,
Replay memory buffer $\mathcal{B}$
Output: Updated actor (*greedy controller*) network π_{ϕ_c}
for $t \leftarrow 1$ **to** *n_crit* **do**
  Sample N transitions (s, a, r, s') from $\mathcal{B}$
  $\epsilon \sim \text{clip}(\mathcal{N}(0, \hat{\sigma}), -c, c)$ // Sample smoothing noise
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi'_c}(s') + \epsilon)$
  $\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum_n (y_n - Q_{\theta i}(s_n, a_n))^2$ // Update critics
  if $t \mod d$ **then**
    // Update greedy controller using gradient descent
    $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_\phi \pi_\phi(s_t) \nabla_a Q_{\theta_1}(s_t, a)\big|_{a=\pi_\phi(s_t)}$
    $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$ // Update target critics
    $\phi'_c \leftarrow \tau\phi_c + (1-\tau)\phi'_c$ // Update target actor
  end if
end for

Chapter 3

BD-aware Critic, Design and Theory

PGA MAP-Elites has successfully integrated Reinforcement Learning tools to scale MAP-Elites to bigger search spaces. Their RL approach however is only concerned with improving solutions of the archive and does not take diversity into account. The Actor-Critic scheme operates in a single-agent setting, with the aim of finding the single best solution to solve a task. This is quite different from the QD objectives we are interested in. In our case, we have a collection of agents that we want to optimise, while at the same time enforcing them to be diverse.

We think that by better adapting the Policy Gradient variation operator we could gain performance. We propose a method to make the PG operator take diversity into account, by using a "behaviour-aware" Critic. In this chapter we state in more details the motivation behind having a BD-aware Critic and explain the design choices made for its training.

3.1 General Idea

3.1.1 Observations

Recent in-depth studies [8; 19] examined the contribution of each variation operator to the archive improvement. They give more insight into the synergies that operate between the GA and the PG operator, from a temporal perspective. Their results show that while the GA operator contributes to archive improvement at all stages of the process, the PG variations only allow adding solutions to the archive at the beginning of the process. This strongly supports the hypothesis of two distinct optimisation phases, as described by [19]. In the early generations, PG variations guide the optimisation process towards promising regions of the search space and ultimately lead to reaching the so-called *elites hyper-volume* introduced by [10]. Just as species on earth share the major part of their genes, this elites hyper-volume is the region of the search space that contains the best solutions. Once the Elites hyper-volume has been discovered, the results from [19] suggest that the GA variations allow to exploit it by an extensive exploration.

These studies make further observations concerning the spatial correlation between the path of the greedy actor and the areas impacted by the PG variations in the behaviour space. We call "greedy actor" the actor that is trained along with the Critic in a classic RL Actor-Critic scheme; it also gets added to the archive after each training phase. [8] was able to visualise the improvement of the archive over time in different replications and observe "greedy paths" along which good solutions emerge. They experimentally show that the PG operator is only able to add solutions to the archive in the "greedy area", which is the one of best global fitness. This observation is in line with the temporal perspective hypothesis stated above. In the first iterations of the algorithm, the greedy actor is driven to the area of global best fitness and the region around it gets optimised efficiently thanks to the PG operator. Once this greedy area has been optimised, the PG operator is unable to suggest good variations in the rest of the space and becomes irrelevant.

To conclude, PGA MAP-Elites's current Policy Gradients operator cannot evolve diverse controllers. We would like the critic to be more general and approximate the optimal action-value function all over the BD space.

3.1.2 A Universal Value Function Approximator

The gradient steps applied to a controller during PG variations have the following effect: making the controller's policy choose, for a given state, actions that maximise the critic's evaluations. However, we raised the issue that such evaluations are behaviour agnostic and in fact drive the controllers to evolve towards the global fitness maximum in the genotype space. To better align with QD objectives, we would like the Critic's evaluations to be dependent on a sought final behaviour or BD *goal*. In the same given state, a certain type of behaviour (e.g. moving towards the left) should not have the same optimal action as another type of behaviour (e.g. moving towards the right).

Recent work [8] proposed to make the values given by the Critic not only depend on a state-action pair but also on the behavioural type of the solution considered. Such Critic would be "aware" of the behaviour descriptor of the controller being evolved. In this work, they added the behaviour descriptor to the Critic network's input but to our knowledge, no work has been done to adapt the critic's training.

In the next Section, we explain how we modified the Critic's training such that its predictions effectively take the behavioural descriptor into account. This new critic training is inspired by Goal-conditioned RL methods, such as *Universal Value Function Approximators* (UVFA) [20]. The motivation behind UVFA is to learn a value function $Q_\theta(s, a, g)$ generalising not only over states s and actions a but also over goals g we may try to achieve. Each goal $g \in \mathcal{G}$ is associated with a corresponding pseudo-reward function $r_g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, fixed for the whole episode, and the corresponding fitness function f_g is simply the discounted sum of such rewards over the course of the trajectory.

Building on these ideas, we consider the behaviour space $\mathcal{B}$ as a goal space $\mathcal{G}$ and would like the Critic to learn a goal-conditioned value function:

$$Q_\theta(s, a, b) = \mathbb{E}[R_t|s_t, a_t, b]$$

To do so, we modify the gradient ascent loss $\mathcal{L}$ used to train the Q network, as further described in section 3.2.2.

3.1.3 BD-parameterised variations

Using this new BD-aware Critic, we can now consider the BD of a solution to be evolved as a goal and condition the Critic's evaluations on this goal. In this new implementation, an actor stores it's behaviour descriptor and gives it to the Critic when a PG variation is applied. The Algorithm 5 describes the new Policy Gradient variations, which will optimise a given solution towards a better fitness, conditioned on the intended behaviour.

Algorithm 5: New PG Variation Operator, with a BD-aware critic

Input : Parent controller to be evolved π_ϕ,
b the behavioural descriptor of the parent controller,
Critic Q_{θ_1},
Experience replay memory buffer $\mathcal{B}$,
Number of gradient update steps n_grad,
Learning rate for gradient variation α

Output: New controller $\pi_{\widehat{\phi}}$

$\widehat{\phi} \leftarrow \phi$

for $i \leftarrow 1$ **to** *n_grad* **do**

Sample N transitions (s, a, r, s') from $\mathcal{B}$

$\nabla_{\widehat{\phi}} J(\widehat{\phi}) = \frac{1}{N} \sum \nabla_{\widehat{\phi}} \pi_{\widehat{\phi}}(s) \nabla_a Q_{\theta_1}(s, a, \mathbf{b})\Big|_{a=\pi_{\widehat{\phi}}(s)}$

$\widehat{\phi} \leftarrow \widehat{\phi} + \alpha \nabla_{\widehat{\phi}} J(\widehat{\phi})$

end for

3.2 A new training method

We would like the critic's values for any possible action to be conditioned on the type of actor considered. We investigate two ideas to re-design the critic training, detailed in this section: using a contrastive loss and using many greedy actors. The overall approach behind BD-aware PGA-MAP-Elites is summarised in Figure 3.1.

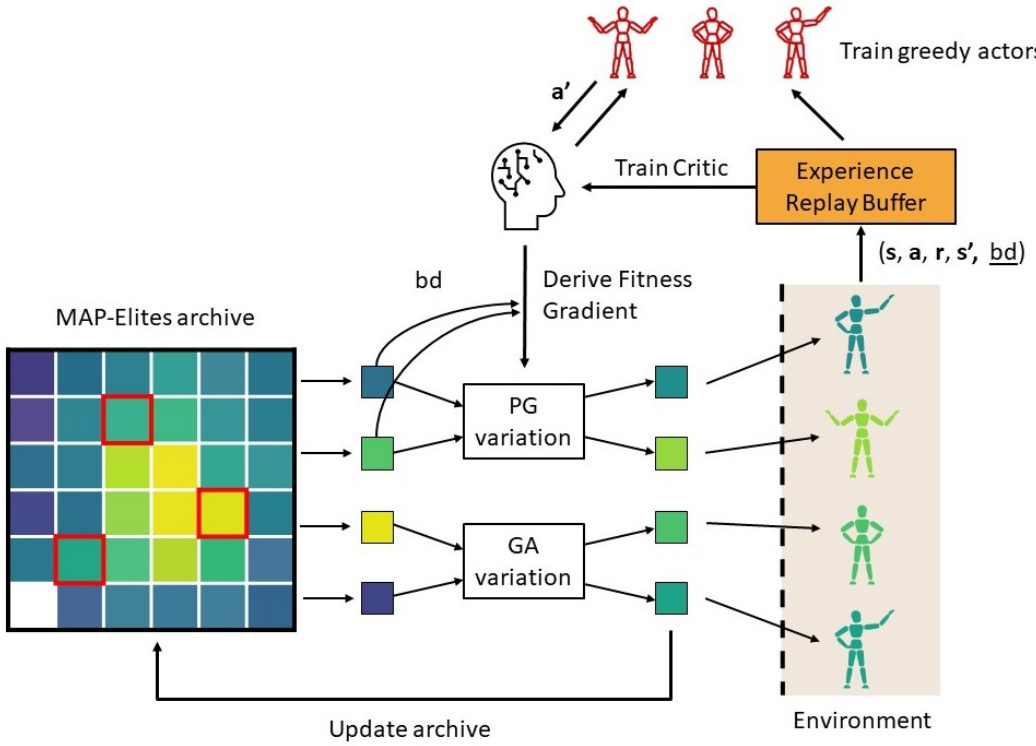


Figure 3.1: Operation scheme of BD-aware PGA-MAP-Elites. A standard MAP-Elites loop is executed repeatedly for selection, variation, evaluation and addition of new solutions to the archive. Variations are equally split between two independent operators: a genetic (GA) variation for divergent search, and a Policy Gradient (PG) variation for directed performance improvement. A BD-aware Critic neural network is trained in parallel, using experience from the evaluations. Greedy actors are trained alongside the critic, they each have their own target BD and act as "local experts" that the Critic can use for its bootstrapped prediction updates. The BD-aware Critic provides performance gradients for the PG operator to evolve solutions with any sought BD. Adapted from [3].

3.2.1 Using many greedy actors

PGA-MAP-Elites uses a single greedy actor in its Actor-Critic scheme. The Critic's TD update uses the classic bootstrapped prediction of future state-action values. The next action for this bootstrap comes from the policy that we are trying to learn or *target policy*. In the PGA MAP-Elites case this is the greedy actor policy. We think this approach has two limitations in a QD context. First, it focuses the Critic's attention on the greedy actor only, with consequences already stated above. The second issue has to do with off-policy learning. With off-policy learning, the behavioural policy used to sample data from the environment should include the one we seek to evaluate, the target policy. The behaviour policy is usually ϵ-greedy with respect to Q^* in RL. In the case of PGA MAP-Elites however, the transitions come from a much broader distribution, for which we have less guaranty. No mechanism such as importance sampling is used to reduce the discrepancy between the data distribution of the policies collecting the data and the one of the target policy. We hypothesise that

this might bias the efficiency of the critic's advice towards solutions that are similar to the greedy actor. Hence, we think that having many regional greedy actors, i.e. one target policy for each region of the behaviour space, could help reducing this discrepancy. The PG operator should suggest relevant evolution to a wider part of $\mathcal{B}$ by using a target Q^* much closer to the policies evolved. Our intuition is to create many of the "greedy paths" observed by [8] in parallel.

How do we go about taking many greedy actors' predictions for the next action in the Critic's update? Preliminary experiments demonstrated that simply initialising different greedy actors and then using the average of their next action prediction in the Critic's bootstrap update will not work. They quickly converge to the same region, and the final Critic's predictions are still optimised for the greedy area only. Hence we keep the idea of using many greedy actors so we can be able to rely on these regional experts for target policies, but use additional mechanisms to prevent them from converging to the same behaviour, which would make the Critic go back to learning a single target policy.

The idea is to train each greedy actor with a respective "target BD" in mind. These target BDs are defined at the beginning of the algorithms, each associated to a different greedy actor, and spread uniformly in the behaviour space to ensure good coverage. Greedy actors are trained alongside the Critic, using the TD3 algorithm just as in the classic PGA MAP-Elites setting. The sensible difference, however, is that during the PG update for a greedy actor π_{ϕ_i}, the aim is to maximise $Q(s, \pi_{\phi_i}(s), target\_bd_i)$: the predictions of the critic conditioned on its target BD $target\_bd_i$. Throughout the training process, while the critic gets better at generalising over states, actions and goals, the greedy actors become better at solving a task, with an intended behaviour.

3.2.2 A contrastive loss

The general idea behind Universal Value Function Approximators, as described in 3.1.2, is to construct goals as parameterised fitness functions. In our case, goals are sampled in the behaviour space. This section explains how we implicitly achieve goal-parameterised fitness functions by weighting the sampled rewards according to distances in the behavioural goal space.

Goals bd_{goal} are randomly sampled in the replay buffer using a "positive sample" $(s_p, a_p, r_p, s'_p, bd_{goal})$, contrasted against a drawn "negative sample" $(s_n, a_n, r_n, s'_n, bd_n)$. For each goal we select the closest greedy actor as a "regional expert" π_{ϕ_g} to derive the next action for the Critic's TD update. The BD-aware Critic learns from the positive sample using the classic TD target from TD3 and the BD goal:

$$y_p = r_p + \gamma \min_{i=1,2} Q_{\theta'_i}\left(s'_p, \pi_{\phi'_g}\left(s'_p\right) + \epsilon, bd_{goal}\right)$$

To learn from the negative sample, we use a pseudo-reward r_g for the BD goal, defined as $r_g = \delta * r_n$: the reward collected by the "contrastive" controller, weighted

by δ according to the distance of this controller to the "goal" controller in the BD space. The contrastive BD target from TD3 becomes:

$$y_n = \delta * r_n + \gamma \min_{i=1,2} Q_{\theta'_i}\left(s'_n, \pi_{\phi'_g}\left(s'_n\right) + \epsilon, bd_{goal}\right).$$

We implement the weight for this goal-conditioned reward as an exponential kernel, where l is the length scale of the kernel.

$$\delta(bd_{goal}, bd_n) = \exp(-distance(bd_{goal}, bd_n)/l)$$

Note that in this new implementation of Critic training, each time a new solution is evaluated in the environment, all transitions are associated with the final BD, and we store every (s, a, r, s', bd) in the replay buffer.

Using the above-described training process, the Critic learns how good a given state-action pair is for a sought behavioural goal. Contrary to the previous setting, the Critic is now able to differentiate that a given state-action pair, although generating high reward and thus being a good transition for a high final fitness, might not be relevant once we consider a particular behaviour. The perfect way of moving a robot's joints in a given configuration to make progress and minimize energy consumption might make the robot move forward and is therefor not so good if we aim to move backwards.

Algorithm 6: Contrastive training for a BD-aware Critic

Input : Critic networks Q_{θ_1}, Q_{θ_2},
Actors networks $\pi_{\phi_{greedy_1}}, ..., \pi_{\phi_{greedy_M}}$,
Target networks $Q_{\theta'_1}$, $Q_{\theta'_1}$, $\pi_{\phi'_{greedy_1}}, ..., \pi_{\phi'_{greedy_M}}$,
Number of critic update steps n_crit,
Sampling batch size N,
Replay memory buffer $\mathcal{B}$,
Frequency of actor update w.r.t. critic training steps $policy\_freq$

Output:

for $t \leftarrow 1$ **to** *n_crit* **do**

Sample N transitions (s, a, r, s', bd) from $\mathcal{B}$

Subset of $N/2$ *positive* samples $(s_p, a_p, r_p, s'_p, bd_{goal})$

Subset of $N/2$ *negative* samples $(s_n, a_n, r_n, s'_n, bd_n)$

// Compute the distance weights with an exponential kernel

$distances = euclidean\_distance(bd_{goal}, bd_n)$

$\delta = \exp(-distances/factor)$

// Select the greedy actors specialized for each bd goal's region

$\phi_{greedy} = select\_closest\_greedy\_actor(bd_{goal})$

// Compute the target Q values with noise $\epsilon \sim \text{clip}(\mathcal{N}(0, \hat{\sigma}), -c, c)$

$$y_p = r_p + \gamma \min_{i=1,2} Q_{\theta'_i}\left(s'_p, \pi_{\phi'_{greedy}}\left(s'_p\right) + \epsilon, bd_{goal}\right).$$

$$y_n = \delta * r_n + \gamma \min_{i=1,2} Q_{\theta'_i}\left(s'_n, \pi_{\phi'_{greedy}}\left(s'_n\right) + \epsilon, bd_{goal}\right).$$

// Update Critic

$\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum[(y_p - Q_{\theta i}(s_p, a_p, bd_{goal}))^2 + (y_n - Q_{\theta i}(s_n, a_n, bd_{goal}))^2]$

if $t \mod$ *policy_freq* **then**

// Update greedy actors ϕ_m and target models

for $m \leftarrow 1$ **to** *num_greedy_actors* **do**

$\nabla_{\phi_m} J(\phi_m) = \frac{1}{N} \sum \nabla_{\phi_m} \pi_{\phi_m}(s) \nabla_a Q_{\theta_1}(s, a, \mathbf{b}_{target\_m})\big|_{a=\pi_{\phi_m}(s)}$

$\phi_m \leftarrow \phi_m + \alpha \nabla_{\phi_m} J(\phi_m)$

$\phi'_m \leftarrow \tau\phi_m + (1-\tau)\phi'_m$

end for

$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$

end if

end for

Chapter 4

Evaluation setup

4.1 Task

The PGA-MAP-Elites paper [3] introduced QDgym [22], a suite of tasks built on OpenAI Gym [21] for QD optimisation benchmarks. All evaluation tasks build on PyBullet [23] robotic locomotion benchmarks, with five different robot designs: `QDWalker`, `QDHalfCheetah`, `QDAnt`, `QDHopper` and `QDHumanoid`. These original tasks use a unidirectional setting, aiming to learn diverse walking gaits that allow to move forward fast. The fitness is defined as the distance travelled by the end of the 1000-step simulation, or walking speed. The behavioural descriptor is defined as the proportion of time each foot is in contact with the ground over the course of the simulation. The BD in thus 4-D in the `QDAnt` task for instance, while it is 2-D in the `QDHumanoid` or `QDWalker` tasks.

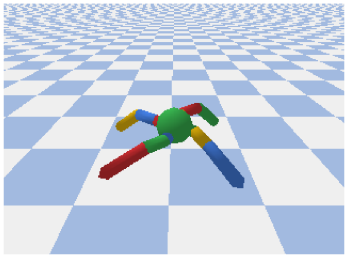


Figure 4.1: `AntBullet-env` from OpenAI Gym [21]

The QDgym_extended [24] package extends these original benchmarks with an omnidirectional setting. The omnidirectional task aims to find efficient ways (minimising energy consumption) of reaching all possible (x, y) positions in a given behaviour space. The fitness of each solution is defined as the sum of an energy usage penalty and a reward for surviving each time-step of the simulation and the BD is defined as the two-dimensional final (x,y) position of the robot.

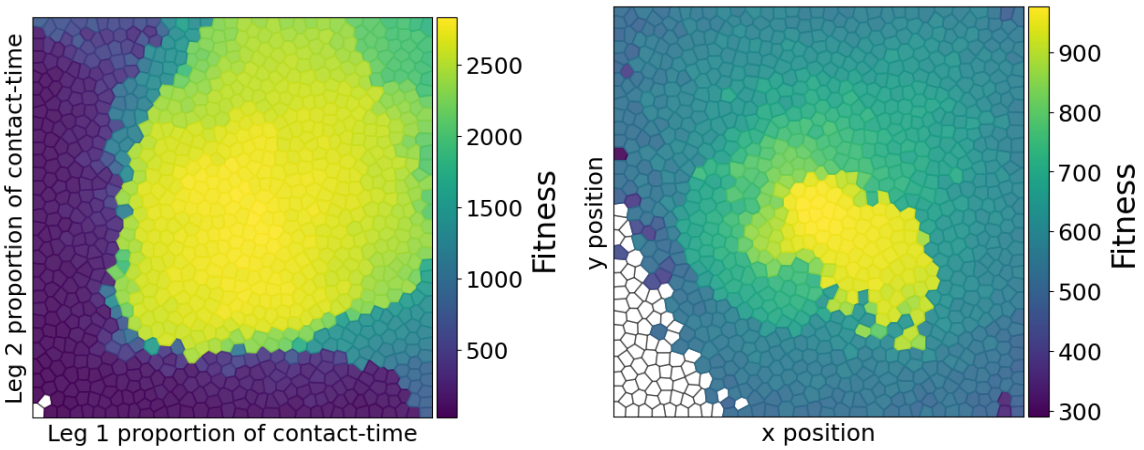
It should be noted that the critic training described in Section 3.2.2 requires all rewards to be positive in order to be weighted by an exponential kernel. We kept the environment's returned rewards unchanged to be able to compare fitness scores to the previous baselines but artificially inflated the rewards stored in the replay buffer by adding a constant term.

We evaluate our work on the `QDDeterministicAntOmnidirectional` task. It is so-called deterministic because the initial state is always the same. Deterministic QDgym tasks set all initial joint positions to 0, whereas they are sampled from a Gaussian

distribution in the original stochastic QDgym tasks. The robot's state has 30 dimensions and is defined as the current height, x, y and z velocity, roll, pitch and yaw angles of the centre of gravity and the relative position of the joints. Each action has 8 dimensions and is defined as the continuous-valued torques applied to each robot joints.

Whilst the original PGA algorithm [3] uses the unidirectional tasks (fitness defined as travelled distance) to evaluate their performance, we decide to use the omnidirectional one (fitness defined as the total energy usage penalty). For this task, optimising the fitness, i.e. saving energy, goes against the diversity objective. Solutions reaching points far from the centre of the behavioural space (where the simulation starts) will necessarily spend more energy, thus will have low fitness and will not be derived by the PG operator. Figure 4.2 illustrates that solutions at the edge of the BD space are poorly optimised and Figure 4.3 shows that the PG operator allows to improve more solutions in `QDWalker` than in `QDDeterministicAntOmnidirectional`. Figure 4.4 also shows that the benefits of adding a PG-operator to MAP-Elites is limited for the Ant omnidirectional task. For theses reasons we think that the omnidirectional setting is of particular interest to test a BD-aware Critic.

Using QD terminology, *genotypes* ϕ are the weights of neural network controllers and *phenotypes* are the trajectories of the robot once evaluated in the environment with the given controller (used to derive the fitness score and the BD). The controller neural networks have the following architecture: an input layer with $state\_dim$ neurons and ReLU activation, two hidden layers with 128 neurons and ReLU activation, and the output layer has $action\_dim$ neurons with tanh activation.



(a) (b)

Figure 4.2: Example of final archives found by PGA MAP-Elites on the `QDWalker` (a) and `QDDeterministicAntOmnidirectional` (b) tasks. The absolute values of fitness should not be compared between tasks as the fitness function is defined differently.

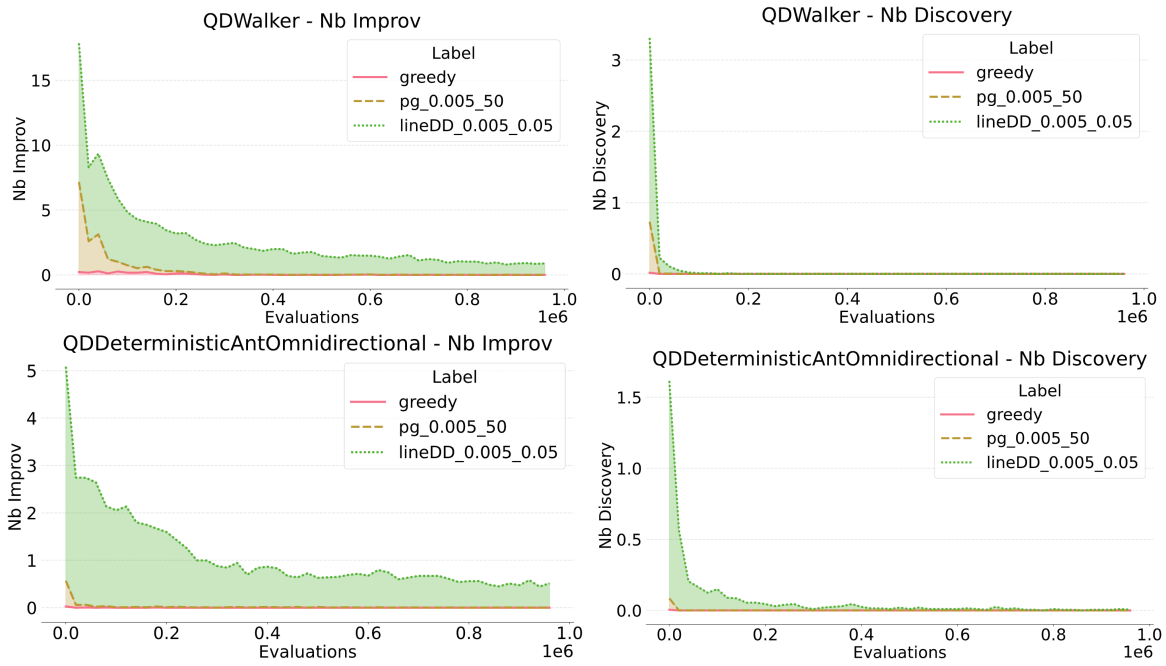


Figure 4.3: Comparison of the variations operators performance in PGA MAP-Elites, on the `QDWalker2DBulletEnv` (left) and `QDDeterministicAntOmnidirectionalBulletEnv` (right) tasks. To measure the contribution of each operator, we use the number of new solutions that each operator added to the archive at each iteration of the algorithm, either by filling a new cell (number of discovery) or by outperforming an existing solution (number of improvement). "pg_0.005_50" refers to the PG operator with a learning rate of 0.005 and 50 steps of gradient ascent. "lineDD_0.005_0.05" refers to the GA operator with a gaussian perturbation σ_1 of 0.005 and a line interpolation noise σ_2 of 0.05 (see 2.1.1). "greedy" refers to the addition of the greedy actor to the archive. Each experiment is averaged over 3 replications with different seeds. We use the code from [25] to generate the plots.

4.2 Metrics

There are various aspects of our method we seek to evaluate. First, we want to compare the general performance of our algorithm against MAP-Elites and PGA MAP-Elites baselines on the Deterministic Ant Omnidirectional task, using metrics commonly used in QD. Diversity is simply measured with the coverage score, which is the number of filled cells in the archive. The quality of the found collection is quantified with several metrics including the median, mean or max fitness of the archive. A popular metric to evaluate QD algorithms is the QD-score, defined as the sum of the fitness of all solutions in the archive, which takes into account both the diversity and the quality of the archive. Monitoring the evolution of all these metrics over time (time should be understood as number of evaluations) allows to study the archive filling dynamics. Additionally, we want to evaluate the performance of the new PG-operator,

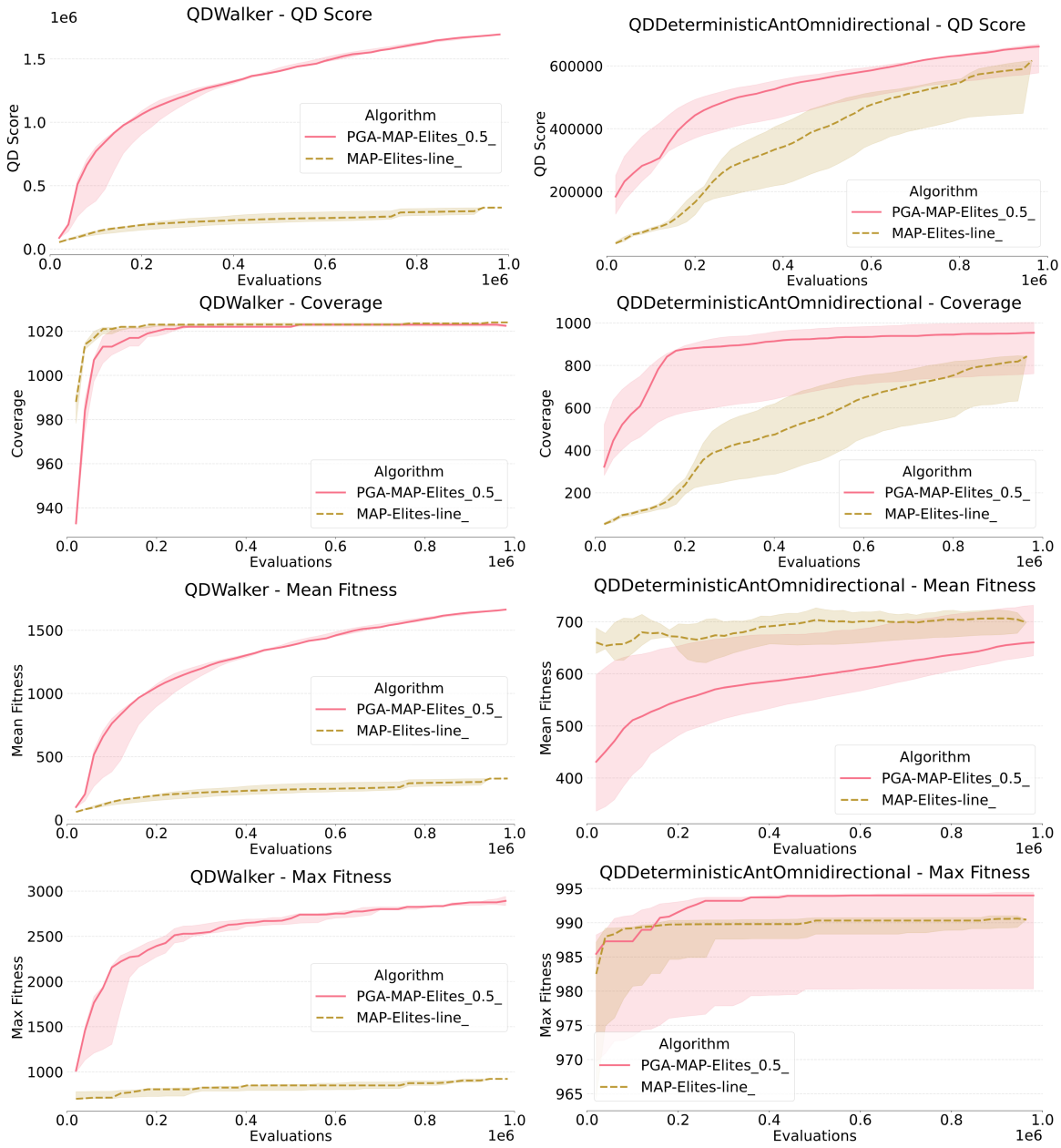


Figure 4.4: Comparison of simple MAP-Elites and PGA MAP-Elites (with 50% of GA variations) performances on `QDWalker` (left) and `QDDeterministicAntOmnidirectional` (right) tasks. The lines represents the median over 3 replications (4 for MAP-Elites) and the shaded area is the interquartile range. The magnitude of the fitness scores should not be compared between tasks as the fitness function is defined differently.

Chapter 5

Results and Evaluation

As a reminder, we propose a new BD-aware critic to be used in the PG variations of PGA MAP-Elites. This new critic is trained using a contrastive loss and many greedy actors in the Actor-Critic scheme. These greedy actors act as "local experts" spread in the behaviour space $\mathcal{B}$. In this section, we first seek to evaluate the individual benefit of using a contrastive loss before moving to an evaluation of the entire scheme, with a varying number of greedy actors.

5.1 Contrastive Learning

To evaluate the benefits of using a contrastive loss in the training of our BD-aware critic, we measure the performance of our method using a single greedy actor. Our results show that by using a single of such "local experts", the performance already significantly improves, reaching the baseline final QD score after only half the number of evaluations (see Fig. 5.1).

A closer examination reveals that this improved QD-score is the results of both a better coverage and a better median fitness, despite a lower best fitness. These metrics jointly describe modified archive filling dynamics, confirmed by the visualisation of the archive on Figure 5.2. Good solutions are more evenly scattered in the grid, which suggests that the "universality" of the new Critic is effective. The median fitness of the archive does not initially grow as fast as in the original setting, instead it grows more steadily and does not plateau, eventually exceeding PGA and correlates with an improved QD-score.

Looking at the max fitness, we see that it is lower with a BD-aware Critic than in the original setting, thus PGA MAP-Elites finds the best solutions. This is not surprising as our contrastive training should slow down the optimisation capability of DRL approaches. The BD-aware critic is learning to generalise not only over state-action pairs but also over behavioural descriptors, which is likely more difficult, whereas the original PG operator can focus on highly optimising the small number solutions around the greedy. Despite this limitation, if we care about the quality of the overall archive as we do in QD, the BD-aware critic still manages to reach a higher QD

score. Furthermore we think that given more computational resources, the max fitness could still improve.

Note that on the Omnidirectional task with a single-greedy setting, the mechanism distributing uniformly the target BDs will give the initial position of the robot as a target BD, which coincidentally corresponds to the fitness global maximum. Although we don't provide an in-depth analysis of the impact of the greedy actor's target BD on the final archive, our analysis with many greedy actors (Section 5.2) and on the QDWalker task (Section 5.2.1) reveal that the performance holds and having a local maximum as a target BD does not seem to greatly impact the algorithm.

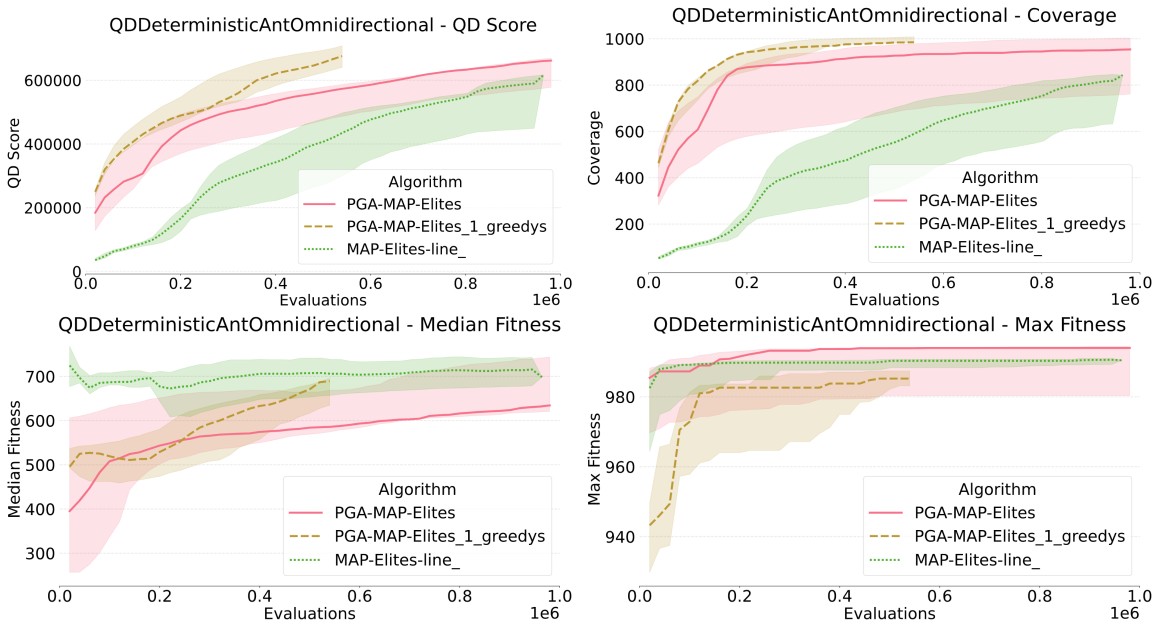


Figure 5.1: MAP-Elites (green), PGA MAP-Elites (red) and our BD-aware Critic PGA, with a single greedy actor (orange). The lines represents the median over 3 replications (4 for MAP-Elites) and the shaded area is the interquartile range. All experiments were run with the same computational resources: 3 days on 32 CPUs.
Median and mean fitness metrics should be considered carefully for comparison as they heavily rely on the size of the archive at each stage.

5.2 Using many actors

We quantify the advantage of having spread-out "local experts" to train the critic by running our method with different numbers of greedy actors. Note that this is not an ablation study purely studying the effect of a Actor-Critic scheme with many Actors, here the Critic is also trained with the contrastive loss already studied above. As already stated in Section 3.2.1 simply initialising different greedy actors will not work; without additional mechanisms they would converge to the global maximum fitness and the algorithm would be back to a classic Actor-Critic scheme.

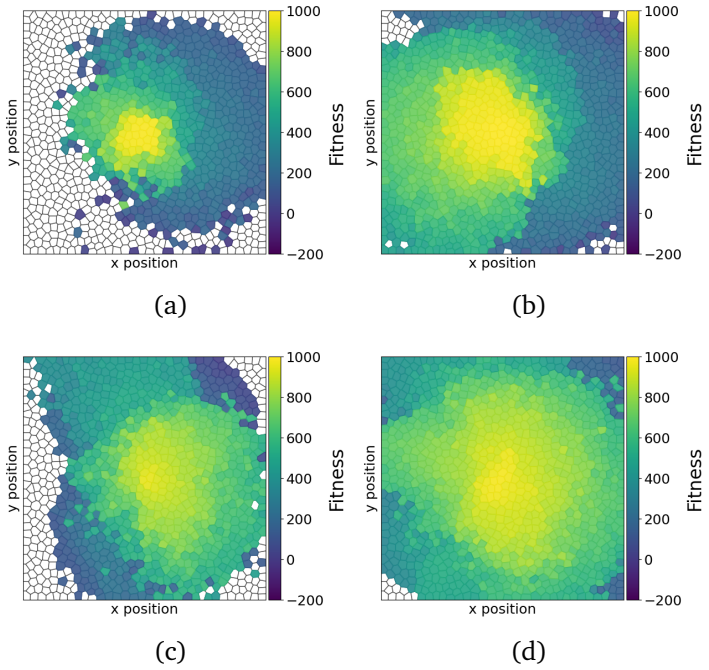


Figure 5.2: Examples of final archives after 1e5 evaluations on the left and 5e5 evaluations on the right (respectively 1:10 and 1:2 of the total evaluations budget) for simple PGA MAP-Elites (top), BD-aware PGA with 1 greedy actor (bottom row).
PGA MAP-Elites:
(a): Sum fitness: 226075, Max fitness: 987, Mean fitness: 372, Median fitness: 280.
(b): Sum fitness: 529900, Max fitness: 994, Mean fitness: 535, Median fitness: 564.
BD-aware PGA MAP-Elites with 1 greedy actor:
(c): Sum fitness: 442171, Max fitness: 953, Mean fitness: 530, Median fitness: 561.
(d): Sum fitness: 673686, Max fitness: 981, Mean fitness: 665, Median fitness: 684.

We consistently observe the same trends as before. The diffusion of good solutions is more homogeneous across the behaviour space as illustrated by the archives in Figure 5.4. We would expect that a higher number of greedy actors allows to cover the behaviour space more quickly as their target BDs get spread further in the behaviour space $\mathcal{B}$. Indeed, discarding the 8-greedy settings, we can observe that in the early generations of the algorithm, around 1e4 evaluations, the coverage score is best for 16 and 4 greedys and gradually decreases for 3, 1 and simple PGA MAP-Elites (see also Fig. 5.4). However the coverage for these high number of greedy actors then plateaus. Instead, the coverage for lower numbers of greedy actors (1, 3) keep growing and reach the maximal coverage by 3e5 evaluations. They also show better median fitness, leading to higher QD-scores (Fig. 5.3). The 1 and 3 greedy actors settings reach the baseline final QD score after only half the number of evaluations.

Ultimately, although giving promising results, the benefits of using many actors is not clear-cut and seems reduced compared to the contrastive training approach. In particular, there is no linear correlation between the number of greedy actors used and the performance of the algorithm (8 greedys does not seem to improve the PGA

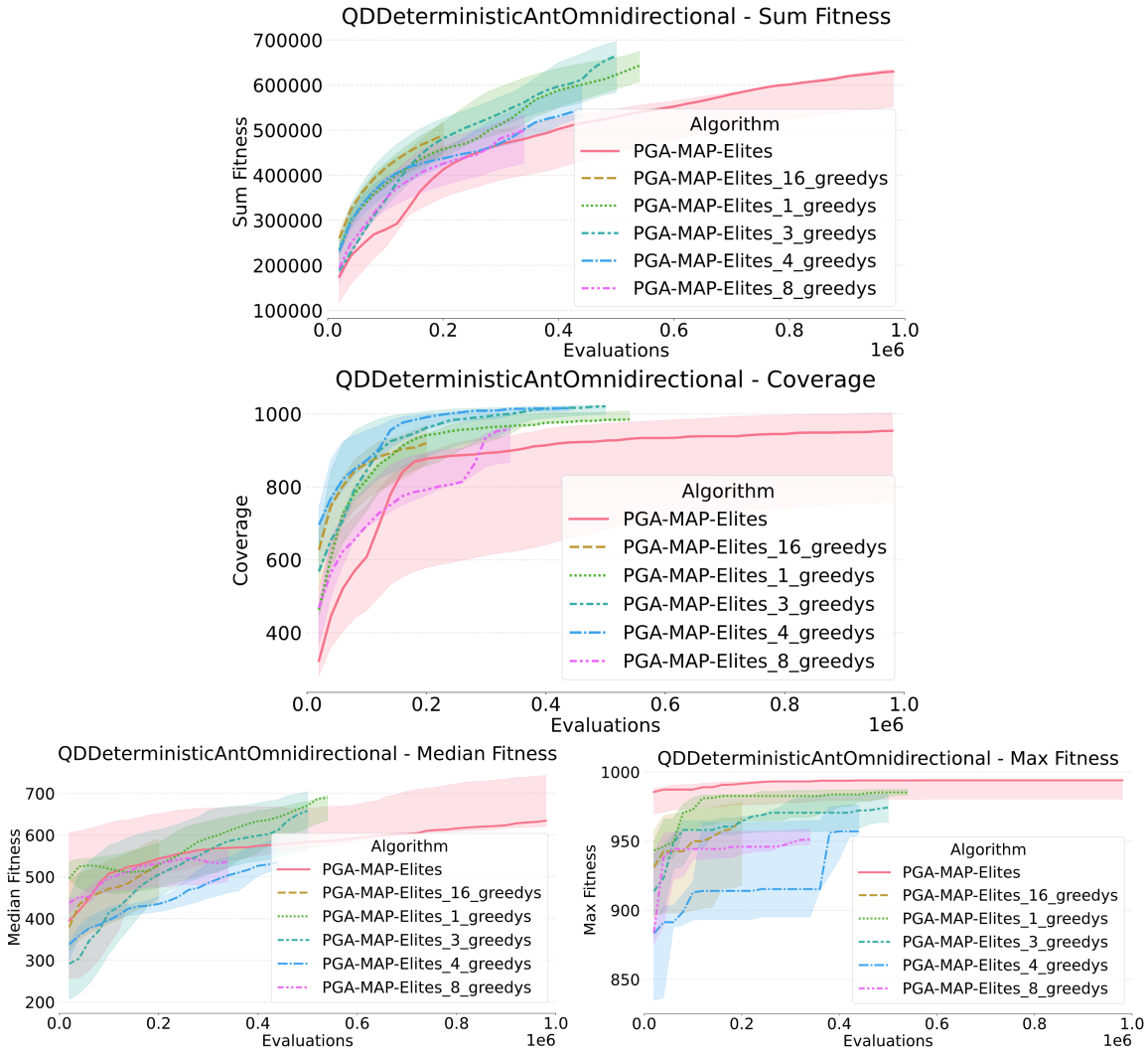


Figure 5.3: Performance of the BD-aware PGA on the Deterministic Ant Omnidirectional task. The lines represents the median over 3 replications and the shaded area is the interquartile range. All experiments were run with the same computational resources: 3 days on 32 CPUs.

MAP-Elites performance while 16 does). Our analysis is limited by the computational constraints that did not allow to reach the same number of total evaluations for all settings nor have a high number of replications. More in-depth analysis would be necessary but because of time constraints, we leave it for future work and focus in the next section on testing our algorithm on a different task.

5.2.1 Additional evaluation on unidirectional task

To evaluate the robustness of our method, we conduct the same experiments on the `QDWalker` task. This task is quite different from the previous deterministic omnidirectional `QDAnt`. First it is unidirectional, meaning that the aim is to learn diverse walking gaits that allow to move forward fast. The fitness is defined as the distance

travelled by the end of the simulation and the 2-D behaviour descriptor is the proportion of time each foot has been in contact with the ground (see Section 4.1). Additionally, the task is stochastic i.e. the initial joints configuration is sampled from a Gaussian and the search space is smaller: actions have 6 dimensions instead of 8 and states have 22 instead of 28.

Despite these differences, our method show similar benefits on this task and we again observe the more even diffusion of good solutions (see Fig. 5.6). As before, the maximum fitness is consistently achieved by simple PGA MAP-Elites, likely because its BD-unaware Critic learns to optimise the greedy area easier and faster than a BD-aware one (small greedy area, no focus on the rest of the space). As opposed to `QDAnt` which has a bigger search space, the maximum coverage (all cells of the grid are filled) is achieved very fast for this task and there is not much space for further improvement. However, the QD-score improves faster thanks to the steeply-rising median fitness (see Fig. 5.5), indicating that our method is able to rapidly find a bigger number of good solutions.

Interestingly, the 4-greedy settings gives the best results for this task, which wasn't the case for `QDDeterministicAntOmnidirectional`, hinting that the optimal number of greedy actors to use might depend on the task. We further discuss this idea in the final chapter.

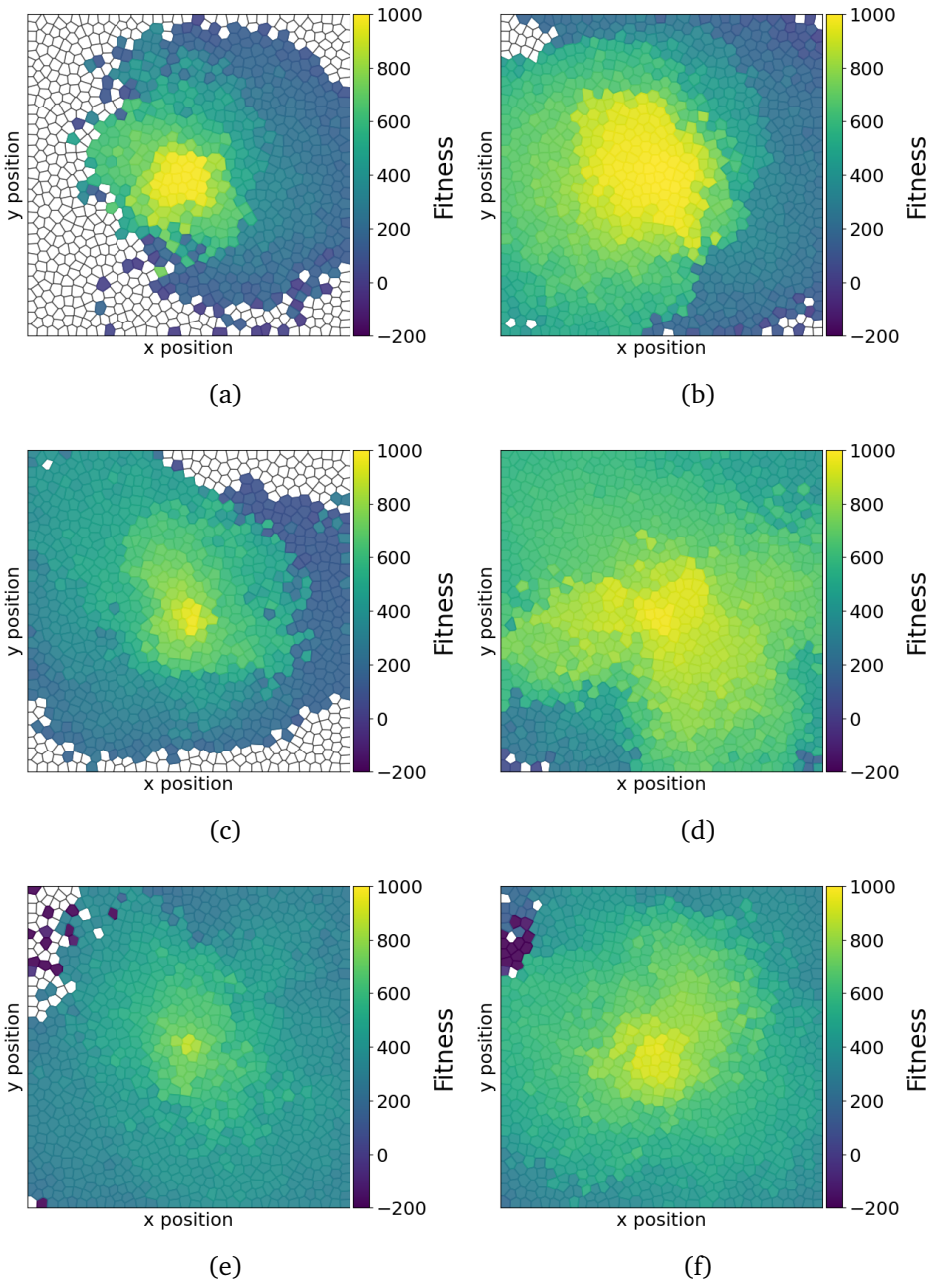


Figure 5.4: Examples of final archives after 1e5 evaluations on the left and 5e5 evaluations (respectively 1:10 and 1:2 of the total evaluations budget) on the right for simple PGA MAP-Elites (top), BD-aware PGA with 3 (middle) and 4 greedy actors (bottom row).
PGA MAP-Elites:
(a): Sum fitness: 226075, Max fitness: 987, Mean fitness: 372, Median fitness: 280.
(b): Sum fitness: 529900, Max fitness: 994, Mean fitness: 535, Median fitness: 564.
BD-aware PGA MAP-Elites with 3 greedy actors:
(c): Sum fitness: 344380, Max fitness: 967, Mean fitness: 407, Median fitness: 415.
(d): Sum fitness: 665413, Max fitness: 974, Mean fitness: 652, Median fitness: 658.
BD-aware PGA MAP-Elites with 4 greedy actors:
(e): Sum fitness: 398470, Max fitness: 886, Mean fitness: 406, Median fitness: 393.
(f): Sum fitness: 540980, Max fitness: 957, Mean fitness: 530, Median fitness: 534.

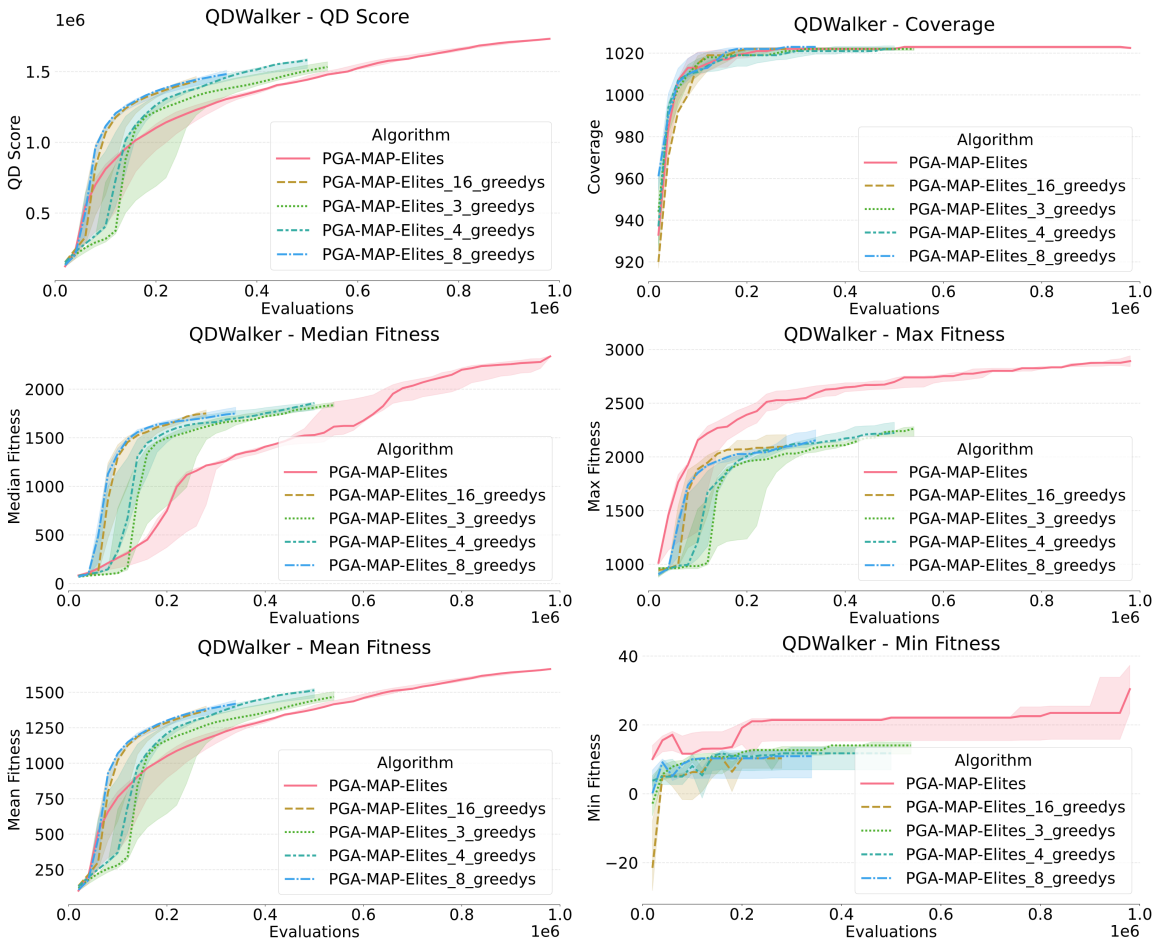
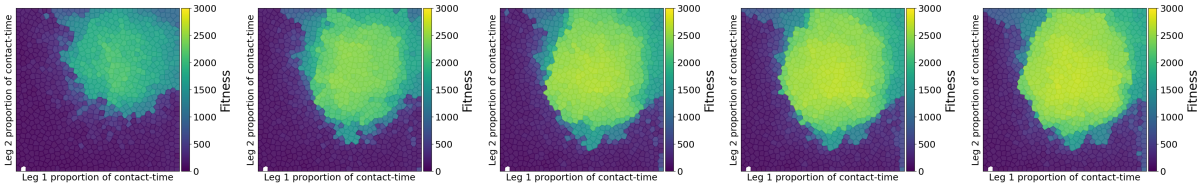
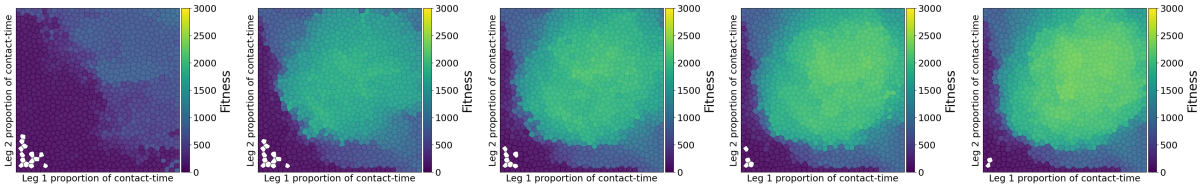


Figure 5.5: Performance of our method against PGA MAP-Elites (red) on the QDWalker task. The lines represents the median over 3 replications and the shaded area is the interquartile range. All experiments were run with the same computational resources: 3 days on 32 CPUs.



(a)



(b)

Figure 5.6: Examples of evolution of the archive for simple PGA MAP-Elites (a) and BD-aware PGA MAP-Elites with 4 greedy actors (b) on `QDWalker`. The snapshots were taken at 1e5, 2e5, 3e5, 4e5 and 5e5 evaluations.

Chapter 6

Conclusions and future work

6.1 Contribution

With this project, we wanted to enhance the use of Reinforcement Learning tools in the Policy Gradient Assisted MAP-Elites algorithm. More precisely, we were concerned with making the PG variation operator take diversity into account and thus proposed to train a BD-aware critic. Our approach revisits the classic Actor-Critic scheme in RL with two contributions and we were able to separately evaluate their benefits.

The first one is a new training method for the critic, using a contrastive loss. This contribution beats the performance of the original PGA MAP-Elites, reaching its final QD-score after only half of the time on the omnidirectional task at study. These results are explained by a change in the archive filling dynamics that we were able to understand through the monitoring of different metrics, confirmed by the visualisation of successive snapshots of the archive. The coverage scores show that our method covers the entire behaviour space more quickly. Moreover, the improved median fitness and the lower best fitness indicate that the distribution of good solutions is more even in the behaviour space.

The second contribution is motivated by the analysis from recent work [8] that shows the PG operator is only able to evolve many good solutions around the path of the greedy actor. Hence, to extend the area of impact of the PG operator we propose to train many greedy actors along with the BD-aware critic and associate to each of them different target BDs spread across the behaviour space. This gave interesting results, although not as apparent as the contrastive loss. The results indicate that the new algorithm is quite robust to the choice of greedy actors as we observed all the numbers we tested seemed to outperform the baseline of the original PGA MAP-Elites (limited by the fact that our experiments could not run for as long due to longer critic training). The improvement of the QD-score is again explained both in terms of an improved median fitness and an improved coverage.

In conclusion, we aimed to better adapt the Reinforcement Learning tools used in

PGA MAP-Elites to the Quality Diversity framework and our results show we were successful. Our results show that the PG operator derived from our new Critic allows a homogeneous diffusion of good solutions across the behaviour space, whereas they were confined to the region of the greedy actor in the previous setting. Based on those fruitful contributions, we are very excited to see further refinements of our contributions and their application. We suggest future directions of research in the following section that we would have liked to investigate if it wasn't for time constraints.

6.2 Future work

Promising results show that our new critic is able to optimise solutions with a sought BD in mind. In our work we used the PG operator as an optimisation tool in which the BD-aware critic is given a parent solution's BD as input to derive BD-parameterised variations. Taking this idea further, we think an exciting direction would be to investigate using the new PG-operator for exploration by giving to the critic the BD for an empty cell instead of the BD of a solution already in the grid.

Additionally, we think a new analysis on how the GA and new PG variations now work together [19; 8] to evolve the population across time and space is necessary. In particular, to optimise the synergies between GA and PG in terms of proportion of parents evolved with each operator at each generation.

Finally, we think that parameter tuning could bring even better results. For instance the learning rates, number of training steps and training batch size we use are the same as the ones used in PGA MAP-Elites and are unlikely to be adapted to a many-greedy-actors setting. The exponential kernel we use for reward weighting in the critic's contrastive learning is also a first proof of concept but more complex options could be investigated.

The parameter tuning could also include the optimal the number of greedy actors to train along with the Critic. Besides the evident computational resources necessary to train them, we suspect that the trade-off to chose the number of greedy actors depends on the dimension of the behaviour space but don't have the evidence to support this intuition. The BD-aware PG scheme would likely benefit from an in-depth analysis of the number of greedy actors' impact and how the number of greedy actors should be chosen for a given environment and task.

References

[1] Cully A, Demiris Y. Quality and Diversity Optimization: A Unifying Modular Framework. IEEE Transactions on Evolutionary Computation. 2018;22(2):245-59. pages i, 2, 3

[2] Mouret JB, Clune J. Illuminating search spaces by mapping elites. 2015. Available from: http://arxiv.org/abs/1504.04909. pages i, 4

[3] Nilsson O, Cully A. Policy Gradient Assisted MAP-Elites. In: The Genetic and Evolutionary Computation Conference. GECCO '21. Lille, France; 2021. Available from: https://hal.archives-ouvertes.fr/hal-03135723. pages i, 2, 12, 13, 17, 21, 22

[4] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. 2nd ed. The MIT Press; 2018. Available from: http://incompleteideas.net/book/the-book-2nd.html. pages 1, 6

[5] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing Atari with Deep Reinforcement Learning. 2013 12. Available from: http://arxiv.org/abs/1312.5602. pages 1

[6] Kilinc O, Hu Y, Montana G. Reinforcement Learning for Robotic Manipulation using Simulated Locomotion Demonstrations. CoRR. 2019;abs/1910.07294. Available from: http://arxiv.org/abs/1910.07294. pages 1

[7] Cully A, Clune J, Tarapore D, Mouret JB. Robots that can adapt like animals. Nature. 2015;521(7553):503-7. Available from: https://doi.org/10.1038/nature14422. pages 1, 4

[8] Chalumeau F. Combining MAP-Elites algorithms with Deep Reinforcement Learning; 2021. Available from: https://gitlab.doc.ic.ac.uk/AIRL/students_projects/2020-2021/felix_chalumeau/report. pages 2, 5, 14, 15, 18, 33, 34

[9] Vassiliades V, Chatzilygeroudis K, Mouret JB. Using Centroidal Voronoi Tessellations to Scale Up the Multidimensional Archive of Phenotypic Elites Algorithm. IEEE Transactions on Evolutionary Computation. 2018;22(4):623-30. Available from: https://doi.org/10.1109/TEVC.2017.2735550. pages 4

[10] Vassiliades V, Mouret JB. Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. In: GECCO 2018 - Genetic and Evolutionary Computation Conference. Kyoto, Japan; 2018. Available from: https://hal.inria.fr/hal-01764739. pages 5, 14

[11] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015 Feb;518(7540):529-33. Available from: http://dx.doi.org/10.1038/nature14236. pages 6, 8

[12] Sutton RS, McAllester D, Singh S, Mansour Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: Solla S, Leen T, Müller K, editors. Advances in Neural Information Processing Systems. vol. 12. MIT Press; 1999. Available from: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf. pages 7

[13] Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning. 1992;8:229-56. pages 7, 10

[14] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al.. Continuous control with deep reinforcement learning. arXiv; 2015. Available from: https://arxiv.org/abs/1509.02971. pages 7, 8

[15] Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic Policy Gradient Algorithms. In: ICML. Beijing, China; 2014. Available from: https://hal.inria.fr/hal-00938992. pages 8

[16] Fujimoto S, van Hoof H, Meger D. Addressing Function Approximation Error in Actor-Critic Methods. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. PMLR; 2018. p. 1587-96. Available from: https://proceedings.mlr.press/v80/fujimoto18a.html. pages 8, 9, 12

[17] Colas C, Madhavan V, Huizinga J, Clune J. Scaling MAP-Elites to Deep Neuroevolution. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. GECCO '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 67–75. Available from: https://doi.org/10.1145/3377930.3390217. pages 10

[18] Pierrot T, Macé V, Chalumeau F, Flajolet A, Cideron G, Beguir K, et al.. Diversity Policy Gradient for Sample Efficient Quality-Diversity Optimization. arXiv; 2020. Available from: https://arxiv.org/abs/2006.08505. pages 11

[19] Anonymous_Author(s). Empirical analysis of PGA-MAP-Elites for Neuroevolution in Stochastic Domains; 2022. Unpublished, under review at TELO. pages 14, 34

[20] Schaul T, Horgan D, Gregor K, Silver D. Universal Value Function Approximators. In: Bach F, Blei D, editors. Proceedings of the 32nd International

Conference on Machine Learning. vol. 37 of Proceedings of Machine Learning Research. Lille, France: PMLR; 2015. p. 1312-20. Available from: https://proceedings.mlr.press/v37/schaul15.html. pages 15

[21] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al.. OpenAI Gym; 2016. Available from: https://github.com/openai/gym. pages 21

[22] Nilsson O. QDgym. GitHub; 2020. Available from: https://github.com/ollenilsson19/QDgym. pages 21

[23] Coumans E, Bai Y. PyBullet, a Python module for physics simulation for games, robotics and machine learning; 2016–2021. Available from: http://pybullet.org. pages 21

[24] Flageat M, Nilsson O. QDgym Extended. GitHub; 2022. Available from: https://github.com/adaptive-intelligent-robotics/QDgym_extended.git. pages 21

[25] Flageat M. Simple PGA MAP-Elites. GitHub; 2021. Available from: https://gitlab.doc.ic.ac.uk/AIRL/research_projects/manon_flageat/meng/simple-pga-map-elites. pages 23