Synergizing Quality-Diversity with Descriptor-Conditioned Reinforcement Learning

ANONYMOUS AUTHOR(S)

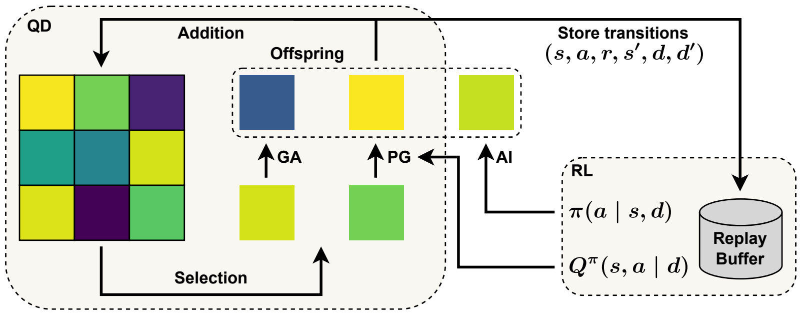


Fig. 1. DCG-MAP-Elites-AI implements a conventional MAP-Elites loop comprising selection, variation, evaluation, addition and leverages two complementary variation operators: a standard Genetic Algorithm (**GA**) variation operator for diversity and a descriptor-conditioned Policy Gradient (**PG**) variation operator for quality. Concurrently to the critic's training, the knowledge of the archive is distilled in the descriptor-conditioned actor. In turn, this versatile actor is injected (**AI**) in the offsprings at each iteration.

A fundamental trait of intelligence involves finding novel and creative solutions to address a given challenge or to adapt to unforeseen situations. Reflecting this, Quality-Diversity optimization is a family of Evolutionary Algorithms, that generates collections of both diverse and high-performing solutions. Among these, MAP-Elites is a prominent example, that has been successfully applied to a variety of domains, including evolutionary robotics. However, MAP-Elites performs a divergent search with random mutations originating from Genetic Algorithms, and thus, is limited to evolving populations of low-dimensional solutions. PGA-MAP-Elites overcomes this limitation using a gradient-based variation operator inspired by deep reinforcement learning which enables the evolution of large neural networks. Although high-performing in many environments, PGA-MAP-Elites fails on several tasks where the convergent search of the gradient-based variation operator hinders diversity. In this work, we present three contributions: (1) we enhance the Policy Gradient variation operator with a descriptor-conditioned critic that reconciles diversity search with gradient-based methods, (2) we leverage the actor-critic training to learn a descriptor-conditioned policy at no additional cost, distilling the knowledge of the population into one single versatile policy that can execute a diversity of behaviors, (3) we exploit the descriptor-conditioned actor by injecting it in the population, despite network architecture differences. Our method, DCG-MAP-Elites-AI, achieves equal or higher QD score and coverage compared to all baselines on seven challenging continuous control locomotion tasks.

CCS Concepts: • **Computing methodologies → Evolutionary robotics**; **Sequential decision making**.

Additional Key Words and Phrases: Quality-Diversity, Reinforcement Learning, Neuroevolution, MAP-Elites, Policy Gradient

ACM Reference Format:
Anonymous Author(s). 2018. Synergizing Quality-Diversity with Descriptor-Conditioned Reinforcement Learning. In . ACM, New York, NY, USA, 30 pages. https://doi.org/XXXXXXX.XXXXXXX

2018. Manuscript submitted to ACM

1 INTRODUCTION

A fascinating aspect of evolution is its ability to generate a variety of different species, each being adapted to their niche. Inspired by this idea, Quality-Diversity (QD) optimization is a family of evolutionary algorithms that aims to generate a set of both high-performing and diverse solutions to a single problem [5, 9, 35]. Contrary to traditional optimization methods that return a single high-performing solution, the goal of QD algorithms is to illuminate a search space of interest called *descriptor space* [30]. Producing a large collection of diverse and effective solutions enables to get multiple alternatives to solve a single problem, which is useful in robotics to improve robustness, recover from damage [8] or reduce the reality gap [6]. Furthermore, conventional optimization methods are prone to get stuck in local optima, whereas keeping a repertoire of diverse solutions to a given problem can help to find stepping stones that lead to globally better solutions [30, 31]. Another benefit of diversity search is efficient exploration in problems where the reward signal is sparse or deceptive [4, 10, 34].

MAP-Elites [30] is a conceptually simple but effective QD optimization algorithm that has shown competitive results in a variety of applications, to generate large collections of diverse skills. However, MAP-Elites relies on random variations that can cause slow convergence in large search spaces [7, 31, 34], making it inadequate to evolve neural networks with a large number of parameters.

In contrast, Deep Reinforcement Learning (RL) [29] algorithms combine reinforcement learning with the directed search power of gradient-based methods in order to learn a single optimal solution. RL has led to remarkable accomplishments in various areas, including in discrete environments like video games [45], board games [39] and in continuous control domains for locomotion [21, 23] and manipulation [32]. These achievements highlight the exceptional capabilities of RL algorithms in addressing specific challenges. Especially, policy gradient methods have shown state-of-the-art results in learning large neural network policies with thousands of parameters in high-dimensional and continuous domains [21, 28, 40].

PGA-MAP-Elites [31] is an extension of MAP-Elites that integrates the sample efficiency of RL algorithms using TD3 [19]. It combines a Policy Gradient (PG) variation operator for efficient fitness improvement, coupled with the usual Genetic Algorithm (GA) variation operator. The PG variation operator leverages gradients derived from RL to drive mutations towards the global fitness optimum and is supported by the divergent search of the GA variation operator for both exploration and optimization [13]. Other recent works have also introduced methods to combine the strength of QD algorithms with reinforcement learning [34, 42] on complex robotics tasks.

PGA-MAP-Elites achieves state-of-the-art performances in most of the environments considered so far in the literature [31, 34, 42]. However, the PG variation operator becomes ineffective in tasks where the global optimum is in an area of the search space that is not likely to produce offspring that are added to the archive. For example, consider a locomotion task where the fitness is the opposite of the energy consumption and the descriptor is defined as the final position of the robot. The global optimum for the fitness is the solution that does not move in order to minimize energy consumption. Thus, the PG variation operator will encourage solutions to stay motionless, collapsing their descriptors to a single point, the descriptor of the global optimum. Consequently, the PG variation operator generates offspring that are discarded and no interesting stepping stone is found, thereby hindering diversity.

DCG-MAP-Elites GECCO [12] builds upon PGA-MAP-Elites algorithm by enhancing the PG variation operator with a descriptor-conditioned critic that provides gradients depending on a target descriptor. The descriptor-conditioned critic takes as input a state and a target descriptor to evaluate actions. Thus, the PG variation operator can mutate solutions to produce offsprings with higher fitness while targeting a desired descriptor, thereby avoiding to collapse their descriptors to a single point.

Furthermore, the descriptor-conditioned critic undergoes training utilizing the RL algorithm TD3 that requires to train an actor in parallel. We take advantage of this intertwined actor-critic training to make the actor 'descriptor-conditioned' as well, allowing it to take actions based not only on the current state but also on a target descriptor we want to achieve. Thus, instead of taking actions that maximize the fitness globally, the actor now takes actions that maximize the fitness while achieving a target descriptor. At the end of training, the result is a versatile agent that can achieve the diversity of behaviors contained in the archive while obtaining similar fitness performance, negating the burden of dealing with a collection of thousands of solutions. In addition to archive distillation, DCG-MAP-Elites GECCO has been shown to improve performance significantly over PGA-MAP-Elites on omnidirectional tasks, while maintaining similar performance on unidirectional tasks where no improvement was expected.

Finally, drawing inspiration from PGA-MAP-Elites that injects the actor in the population at each generation, we extend the original DCG-MAP-Elites GECCO version [12] with a descriptor-conditioned Actor Injection (AI), that enables to inject the versatile actor in the population, despite network architecture differences.

In summary, we introduce DCG-MAP-Elites-AI (Descriptor-Conditioned Gradients MAP-Elites with Actor Injection) that extends DCG-MAP-Elites GECCO and present three contributions: (1) we enhance the PG variation operator with a descriptor-conditioned critic, (2) we distill the knowledge of the archive into one single versatile policy at no additional cost, (3) we take advantage of this high-performing and versatile policy to improve the population during training with actor injection, further improving our method. We compare our algorithm to four state-of-the-art QD algorithms on seven challenging continuous control locomotion tasks. Our method, DCG-MAP-Elites-AI, achieves equal or higher QD score and coverage compared to all baselines on seven challenging continuous control locomotion tasks.

2 BACKGROUND

2.1 Problem Statement

We consider an agent sequentially interacting with an environment at discrete time steps t for an episode of length T. At each time step t, the agent observes a state s_t, takes an action a_t and receives a scalar reward r_t. We model it as a Markov Decision Process (MDP) which comprises a *state space* $\mathcal{S}$, a continuous *action space* $\mathcal{A}$, a stationary *transition dynamics distribution* $p(s_{t+1} \mid s_t, a_t)$ and a *reward function* $r \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In this work, a *policy* (also called *solution*) is a deterministic neural network parameterized by $\phi \in \Phi$, and denoted $\pi_\phi \colon \mathcal{S} \to \mathcal{A}$. The agent uses its policy to select actions and interact with the environment to give a trajectory of states, actions and rewards. The *fitness* of a solution is given by $F \colon \Phi \to \mathbb{R}$, defined as the expected discounted return $\mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right]$.

In this setting, the objective of QD algorithms is to find the highest fitness solutions in each point of the *descriptor space* $\mathcal{D}$. The descriptor function $D \colon \Phi \to \mathcal{D}$ is generally defined by the user and characterizes solutions in a meaningful way for the type of diversity desired. With this notation, our objective is to evolve a population of solutions that are both high-performing with respect to F and diverse with respect to D.

2.2 MAP-Elites

Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [30] is a simple yet effective QD algorithm, that discretizes the descriptor space $\mathcal{D}$ into a multi-dimensional grid of cells called archive $\mathcal{X}$ and searches for the best solution in each cell, see Algorithm 14. The goal of the algorithm is to return an archive that is filled as much as possible with high-fitness solutions. MAP-Elites starts by generating random solutions and adding them to the archive. The algorithm then repeats the following steps until a budget of I solutions have been evaluated: (1) a batch of solutions from the archive are uniformly selected and modified through mutations and/or crossovers to produce offspring, (2) the fitnesses and descriptors of the offspring are evaluated, and each offspring is placed in its corresponding cell if and only if the cell is empty or if the offspring has a better fitness than the current solution in that cell, in which case the current solution is replaced. As most evolutionary methods, MAP-Elites relies on undirected updates that are agnostic to the fitness objective. With a Genetic Algorithm (GA) variation operator, MAP-Elites performs a divergent search that may cause slow convergence in high-dimensional problems due to a lack of directed search power, and thus, is performing best on low-dimensional search space [31].

2.3 Deep Reinforcement Learning

Deep Reinforcement Learning (RL) [29] combines the reinforcement learning framework with the function approximation capabilities of deep neural networks to represent policies and value functions in high-dimensional state and action spaces. In opposition to black-box optimization methods like evolutionary algorithms, RL leverages the structure of the MDP in the form of the Bellman equation to achieve better sample efficiency. The objective is to find an optimal policy π_ϕ, which maximizes the expected return or fitness $F(\pi_\phi)$. In reinforcement learning, many approaches try to estimate the action-value function $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{i=0}^{T-t-1} \gamma^i r_{t+i} \mid s_t = s, a_t = a\right]$ defined as the expected discounted return starting from state s, taking action a and thereafter following policy π.

The Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [19] is an actor-critic, off-policy reinforcement learning method that achieves state-of-the-art results in environments with large and continuous action space. TD3 indirectly learns a policy π_ϕ via maximization of the action-value function $Q_\theta(s,a)$. The approach is closely connected to Q-learning [19] and tries to approximate the optimal action-value function $Q^*(s,a)$ in order to find the optimal action $\pi^*(s) = \arg\max_a Q^*(s,a)$. However, computing the maximum over action in $\max_a Q_\theta(s,a)$ is intractable in continuous action space, hence it is approximated using $\max_a Q_\theta(s,a) = Q_\theta(s,\pi_\phi(s))$. In TD3, the policy π_ϕ takes actions in the environment and the transitions are stored in a replay buffer. The collected experience is then used to train a pair of critics $Q_{\theta_1}, Q_{\theta_2}$ using temporal difference. Target networks $Q_{\theta_1'}, Q_{\theta_2'}$ are updated to slowly track the main networks. Both critics use a single regression target y, calculated using whichever of the two target critics gives a smaller estimated value and using target policy smoothing by sampling a noise $\epsilon \sim \text{clip}(\mathcal{N}(0,\sigma), -c, c)$:

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

Both critics are learned by regression to this target and the policy is learned with a delay, only updated every Δ iterations simply by maximizing Q_{θ_1} with $\max_\phi \mathbb{E}\left[Q_{\theta_1}(s, \pi_\phi(s))\right]$. The actor is updated using the deterministic policy gradient:

$$\nabla_\phi J(\phi) = \mathbb{E}\left[\nabla_\phi \pi_\phi(s) \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)}\right] \tag{2}$$

2.4 PGA-MAP-Elites

Policy Gradient Assisted MAP-Elites (PGA-MAP-Elites) [31] is an extension of MAP-Elites that is designed to evolve deep neural networks by combining the directed search power and sample efficiency of RL methods with the exploration capabilities of genetic algorithms, see Algorithm 9. The algorithm follows the usual MAP-Elites loop of selection, variation, evaluation and addition for a budget of I iterations, but uses two parallel variation operators: half of the offspring are generated using a standard Genetic Algorithm (GA) variation operator and half of the offspring are generated using a Policy Gradient (PG) variation operator. During each iteration of the loop, PGA-MAP-Elites stores the transitions from offspring evaluation in a replay buffer $\mathcal{B}$ and uses it to train a pair of critics based on the TD3 algorithm, described in Algorithm 10. The trained critic is then used in the PG variation operator to update the selected solutions from the archive for m gradient steps to select actions that maximize the approximated action-value function, as described in Algorithm 11. At each iteration, the critics are trained for n steps of gradients descents towards the target described in Equation (1), averaged over N transitions of experience sampled uniformly from the replay buffer $\mathcal{B}$. The actor learns with a delay Δ via maximization of the critic according to Equation (2).

3 RELATED WORK

3.1 Scaling QD to Neuroevolution

The challenge of evolving diverse solutions in a high-dimensional search space has been an active research subject over recent years. MAP-Elites-ES [7] scales MAP-Elites to high-dimensional solutions parameterized by large neural networks. This algorithm leverages Evolution Strategies [36] (ES) to perform a directed search that is more efficient than random mutations used in Genetic Algorithms. Fitness and novelty gradients are estimated locally from many perturbed versions of the parent solution to generate a new one. The population tends towards regions of the parameter space with higher fitness or novelty but it requires to sample and evaluate a large number of solutions, making it particularly data inefficient. To improve sample efficiency, methods that combine MAP-Elites with RL [31, 33, 34, 42] have emerged and use time step level information to efficiently evolve populations of high-performing and diverse neural network for complex tasks. PGA-MAP-Elites [31] uses policy gradients for part of its mutations, see Section 2.4 for details. CMA-MEGA [42] estimates descriptor gradients with ES and combines the fitness gradient and the descriptor gradients with a CMA-ES mechanism [16, 22]. QD-PG [34] introduces a diversity reward based on the novelty of the states visited and derives a policy gradient for the maximization of those diversity rewards which helps exploration in settings where the reward is sparse or deceptive. PBT-MAP-Elites [33] mixes MAP-Elites with a population based training process [25] to optimize hyper-parameters of diverse RL agents. Interestingly, recent work [41] scales the algorithm CMA-MAE [17] to high-dimensional policies on robotics tasks with pure ES while showing comparable data efficiency to QD-RL approaches, but is still outperformed by PGA-MAP-Elites.

3.2 Conditioning the critic

None of the methods described in the previous section take a descriptor into account when deriving policy gradients used to mutate solutions. In other words, they do not use descriptor-conditioned policies nor descriptor-conditioned critics as our method does. The concept of descriptor-conditioned critic is related to Universal Value Function Approximators [37], extensively used in skill discovery reinforcement learning, a field that share a similar motivation to QD [2]. In VIC, DIAYN, DADS, SMERL [11, 20, 27, 38], the actors and critics are conditioned on a sampled prior but does not correspond to a real posterior like in DCG-MAP-Elites-AI. Furthermore, those methods use a notion of diversity defined at the step-level rather than trajectory-level like DCG-MAP-Elites-AI. Moreover, they do not use an archive to store a population, resulting in much smaller sets of final policies. Finally, it has been shown that QD methods are competitive with skill discovery reinforcement learning algorithms [2], specifically for adaptation and hierarchical learning.

3.3 Archive distillation

Distilling the knowledge of an archive into a single policy is an alluring process that reduces the number of parameters outputted by the algorithm and enables generalization and interpolation/extrapolation. Although distillation is usually referring to policy distillation — learning the observation/action mapping from a teacher policy — we present archive distillation as a general term referring to any kind of knowledge transfer from an archive to another model, should it be the policies, transitions experienced in the environment, full trajectories or discovered descriptors.

To the best of our knowledge, only two QD-related works use the concept of archive distillation. Go-Explore [10] keeps an archive of states and trains a goal-conditioned policy to reproduce the trajectory of the policy that reached that state. Another related approach is to learn a generative policy network [26] over the policies contained in the archive. Our approach DCG-MAP-Elites-AI distills the experience of the archive into a single versatile policy.

4 METHODS

Algorithm 1 DCG-MAP-Elites-AI

Require: GA batch size b_{GA}, PG batch size b_{PG}, Actor Injection batch size b_{AI}, total batch size $b = b_{\text{GA}} + b_{\text{PG}} + b_{\text{AI}}$

```
Initialize archive 𝒳 with b random solutions and replay buffer ℬ
Initialize critic networks Q_θ1, Q_θ2 and actor network π_φ
i ← 0
while i < I do
    TRAIN_ACTOR_CRITIC(π_φ, Q_θ1, Q_θ2, ℬ)
    π_ψ1, ..., π_ψb ← SELECTION(𝒳)
    π_ψ̂1, ..., π_ψ̂bGA ← VARIATION_GA(π_ψ1, ..., π_ψbGA)
    π_ψ̂bGA+1, ..., π_ψ̂bGA+bPG ← VARIATION_PG(π_ψbGA+1, ..., π_ψbGA+bPG, Q_θ1, ℬ)
    π_ψ̂bGA+bPG+1, ..., π_ψ̂b ← ACTOR_INJECTION(π_φ)
    ADDITION(π_ψ̂1, ..., π_ψ̂b, 𝒳, ℬ)
    i ← i + b
function ADDITION(π_ψ̂ ..., 𝒳, ℬ)
    for π_ψ̂ ... do
        (f, transitions) ← F(π_ψ̂), d ← D(π_ψ̂)
        INSERT(ℬ, transitions)
        if 𝒳(d) = ∅ or F(𝒳(d)) < f then
            𝒳(d) ← π_ψ̂
```

Our method Descriptor-Conditioned Gradients MAP-Elites with Actor Injection (DCG-MAP-Elites-AI) overcomes the limitations of PGA-MAP-Elites by leveraging a descriptor-conditioned critic to improve the PG variation operator and concurrently distills the knowledge of the archive in a single versatile policy as a by-product of the actor-critic training. The pseudocode is provided in Algorithm 1. The algorithm follows the usual MAP-Elites loop of selection, variation, evaluation and addition for a budget of I iterations. Two complementary and independent variation operators are used in parallel: (1) a standard GA operator (2) a descriptor-conditioned PG operator. At each iteration, the transitions from the evaluation step are stored in a replay buffer and used to train an actor-critic pair based on TD3.

Contrary to PGA-MAP-Elites, the actor-critic pair is descriptor-conditioned. In addition to the state s and action a, the critic $Q_\theta\,(s, a \mid d)$ also depends on the descriptor d and estimates the expected discounted return starting from state

s, taking action a and thereafter following policy π *and* achieving descriptor d. In this work, to achieve descriptor d means that the trajectory generated by the policy π has descriptor d. In addition to the state s, the actor $\pi_\phi(s \mid d)$ also depends on a target descriptor d and maximizes the expected discounted return conditioned on achieving the target descriptor d. Thus, the goal of the descriptor-conditioned actor is to achieve the desired descriptor d *while* maximizing fitness.

4.1 Descriptor-Conditioned Critic

Instead of estimating the action-value function with $Q_\theta(s, a)$, we want to estimate the descriptor-conditioned action-value function with $Q_\theta(s, a \mid d)$. When a policy π interacts with the environment, it generates a trajectory, which is a sequence of transitions (s, a, r, s') with descriptor d. We extend the definition of a transition (s, a, r, s') to include the observed descriptor d of the trajectory (s, a, r, s', d). However, the descriptor is only available at the end of the episode, therefore the transitions can only be augmented with the descriptor after the episode is completed. In all the tasks we consider, the reward function is positive $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ and hence, the fitness function F and action-value function are positive as well. Thus, for any target descriptor $d' \in \mathcal{D}$, we define the descriptor-conditioned critic as equal to the normal action-value function when the policy achieves the target descriptor d' and as equal to zero when the policy does not achieve the target descriptor d'. Given a transition (s, a, r, s', d), and a target descriptor d' sampled in $\mathcal{D}$,

$$Q_\theta(s, a \mid d') := \begin{cases} Q_\theta(s, a), & \text{if } d = d' \\ 0, & \text{if } d \neq d' \end{cases} \tag{3}$$

However, with this piecewise definition, the descriptor-conditioned action-value function is not continuous and violates the universal approximation theorem continuity hypothesis [24]. To address this issue, we introduce a similarity function $S: \mathcal{D}^2 \to]0, 1]$ defined as $S(d, d') = e^{-\frac{\|d-d'\|_\mathcal{D}}{l}}$ to smooth the descriptor-conditioned critic and relax Equation (3) into:

$$\begin{aligned} Q_\theta\left(s, a \mid d'\right) = S(d, d')\, Q_\theta(s, a) &= S(d, d')\, \mathbb{E}_\pi\left[\sum_{i=0}^{T-t-1} \gamma^i r_{t+i} \middle| s, a\right] \\ &= \mathbb{E}_\pi\left[\sum_{i=0}^{T-t-1} \gamma^i S(d, d') r_{t+i} \middle| s, a\right] \end{aligned} \tag{4}$$

With Equation (4), we demonstrate that learning the descriptor-conditioned critic is equivalent to scaling the reward by the similarity $S(d, d')$ between the descriptor of the trajectory d and the target descriptor d'. Therefore, the critic target in Equation (1) is modified to include the similarity scaling and the descriptor-conditioned actor:

$$y = S(d, d')\, r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, \pi_{\phi'}(s_{t+1} \mid d') + \epsilon \mid d') \tag{5}$$

If the target descriptor d' is approximately equal to the observed descriptor d of the trajectory $d \approx d'$, then we have $S(d, d') \approx 1$ so the reward is unchanged. However, if the descriptor d' is different from the observed descriptor d, then the reward is scaled down to $S(d, d')\, r(s_t, a_t) \approx 0$. The scaling ensures that the magnitude of the reward depends not only on the quality of the action a with regards to the fitness function F, but also on achieving the target descriptor d'. Given one transition (s, a, r, s', d), we can generate infinitely many critic updates by sampling a target descriptor $d' \in \mathcal{D}$. This is leveraged in the new actor-critic training introduced with DCG-MAP-Elites-AI, which is detailed in Algorithm 2 and Section 4.3.

4.2 Descriptor-Conditioned Actor and Archive Distillation

The training of the critic requires to train an actor π_ϕ to approximate the optimal action a^*, as explained in Section 2.3. However, in this work, the action-value function estimated by the critic is conditioned on a descriptor d. Hence, we don't want π_ϕ to estimate the best action globally, but rather the best action given that it achieves the target descriptor d. Therefore, the actor is extended to a descriptor-conditioned policy $\pi_\phi(s \mid d)$, that maximizes the descriptor-conditioned critic's value with $\max_\phi \mathbb{E}\left[Q_\theta(s, \pi_\phi(s \mid d) \mid d)\right]$. The actor is updated using the deterministic policy gradient, see Algorithm 2:

$$\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_\phi \pi_\phi(s \mid d') \nabla_a Q_{\theta_1}(s, a \mid d')|_{a=\pi_\phi(s|d')} \tag{6}$$

The policy $\pi_\phi(s \mid d)$ learns to suggest actions a that optimize the return *while* generating a trajectory achieving descriptor d. Consequently, the descriptor-conditioned actor can exhibit a wide range of descriptors, effectively distilling some of the capabilities of the archive into a single versatile policy.

4.3 Actor-Critic Training

Algorithm 2 Descriptor-conditioned Actor-Critic Training

> **function** TRAIN_ACTOR_CRITIC($\pi_\phi, Q_{\theta_1}, Q_{\theta_2}, \mathcal{B}$)
>   **for** $t = 1 \rightarrow n$ **do**
>   Sample N transitions (s, a, r, s', d, d') from $\mathcal{B}$
>   Sample smoothing noise ϵ
> $y \leftarrow S(d, d')\, r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi'}(s' \mid d') + \epsilon \mid d')$
>   Update both critics by regression to y
>   **if** t mod Δ **then**
>   Update actor using the deterministic policy gradient:
> $\frac{1}{N} \sum \nabla_\phi \pi_\phi(s \mid d') \nabla_a Q_{\theta_1}(s, a \mid d')|_{a=\pi_\phi(s|d')}$
>   Soft-update target networks $Q_{\theta i'}$ and $\pi_{\phi'}$

In Section 4.1, we show that the descriptor-conditioned critic target y in Equation (5) requires a transition (s, a, r, s', d) and a target descriptor d'. Most related methods that are conditioned on skills or goals rely on a sampling strategy. For example, HER [1] is a goal-conditioned reinforcement learning algorithm that relies on a handcrafted goal sampling strategy and DIAYN, DADS, SMERL sample skills from a uniform prior distribution. However, in this work, we don't need to rely on an explicit descriptor sampling strategy.

For each PG variation operator offspring, the transitions coming from the evaluation step, are populated with d' equal to the descriptor of the parent solution d_ψ. The PG variation operator mutates the parent to improve fitness while achieving descriptor d_ψ. Thus, although the offspring is not descriptor-conditioned, its implicit target descriptor is d_ψ. Consequently, we set the target descriptor d' to the descriptor of the parent d_ψ.

Similarly, for each GA variation operator offspring, the transitions coming from the evaluation step, are populated with d' equal to the observed descriptor of the trajectory d. The GA variation operator mutates the parent by adding random noise to the genotype. However, a small random change in the parameters of the parent solution can induce big changes in the behavior of the offspring, making them behaviorally different. Consequently, we set the target descriptor d' to the observed descriptor of the trajectory d.

At the end of the evaluation step, we augment the transitions with the observed descriptor of the trajectory d, and with the target descriptor d', using the implicit descriptor sampling strategy explained above, giving (s, a, r, s', d, d').

This implicit descriptor sampling strategy has two benefits. First, half of the transitions have $d = d'$, providing the actor-critic training with samples where the target descriptor is achieved, therefore alleviating sparse reward problems. Second, at the beginning of the training process, half of the transitions will have $d \neq d'$ because the solutions in the archive have not learned to accurately achieve their descriptors yet. However, as training goes on, the number of samples where the descriptor is not achieved will decrease, providing some kind of automatic curriculum. Finally, the actor-critic training is adapted from TD3 and is given in Algorithm 2.

4.4 Descriptor-Conditioned PG Variation

Algorithm 3 Descriptor-conditioned PG Variation

function VARIATION_PG($\pi_\psi \dots, Q_{\theta_1}, \mathcal{B}$)
  for $\pi_\psi \dots$ **do**
    $d_\psi \leftarrow D(\pi_\psi)$
    for $i = 1 \rightarrow m$ **do**
      Sample N transitions (s, a, r, s', d, d') from $\mathcal{B}$
      Update actor using the deterministic policy gradient:
      $\frac{1}{N} \sum \nabla_\psi \pi_\psi(s) \nabla_a Q_{\theta_1}(s, a \mid d_\psi)|_{a=\pi_\psi(s)}$
  return $\pi_{\widehat{\phi}} \dots$

Once the critic $Q_\theta(s, a \mid d)$ is trained, it can be used to improve the fitness of any solutions in the archive, as described in Algorithm 3. First, a parent solution π_ψ is selected from the archive and we denote its descriptor by $d_\psi := D(\pi_\psi)$. Notice that this policy $\pi_\psi(s)$ is not descriptor-conditioned, contrary to the actor $\pi_\phi(s \mid d)$. Second, we apply the PG variation operator from Equation (7), for m gradient steps, using the descriptor d_ψ to condition the critic:

$$\nabla_\psi J(\psi) = \frac{1}{N} \sum \nabla_\psi \pi_\psi(s) \nabla_a Q_{\theta_1}(s, a \mid d_\psi)|_{a=\pi_\psi(s)} \tag{7}$$

The goal is to improve the quality of the solution π_ψ, while keeping the same diversity d_ψ. To that end, the critic is used to evaluate actions and guides π_ψ to (1) improve fitness, while (2) achieving descriptor d_ψ.

4.5 Descriptor-Conditioned Actor Injection

Algorithm 4 Descriptor-conditioned Actor Injection

function ACTOR_INJECTION(π_ϕ)
  $d_1, \dots, d_{b_{\text{AI}}} \sim \mathcal{U}(\mathcal{D})$
  $\psi_1, \dots, \psi_{b_{\text{AI}}} \leftarrow$ PARAMETERS_RECOMPUTATION($\pi_\phi(. \mid d_1), \dots, \pi_\phi(. \mid d_{b_{\text{AI}}})$)
  return $\pi_{\psi_1}, \dots, \pi_{\psi_{b_{\text{AI}}}}$

In PGA-MAP-Elites, the actor is injected in the offsprings and considered for addition in the archive at each generation. Empirical analyses [13] have demonstrated the importance of actor injection to achieve good performance. Similarly to PGA-MAP-Elites, we devise a descriptor-conditioned actor injection (AI) mechanism, to improve the performance of our method, DCG-MAP-Elites-AI.

There is however a significant challenge. The GA isoline variation operator [44] used in PGA-MAP-Elites and DCG-MAP-Elites GECCO requires that all policies in the archive share the same architecture. However, in DCG-MAP-Elites-AI, the actor is descriptor-conditioned, while the policies in the archive are not. Thus, the first layer of the actor is larger because it takes as input a state and a descriptor, while the first layer of the policies in the archive are smaller because they take as input only a state. Specifically, for the first layer of the policies in the archive, the weights are a matrix of dimension $(\dim(\mathcal{S}), 128)$ and the biases are a vector of dimension 128. In contrast, for the first layer of the descriptor-conditioned actor, the weights are a matrix of dimension $(\dim(\mathcal{S}) + \dim(\mathcal{D}), 128)$ and the biases are a vector of dimension 128. In both cases, the first hidden layer has 128 neurons, and the subsequent layers are the same.

However, for a given fixed descriptor d, we can consider that the constant descriptor d, in $\pi_\phi(s \mid d)$ is not part of the input, but part of the parameters. As a matter of fact, for a static descriptor d, we can obtain an equivalent specialized policy $\pi_{\psi_d}(s)$ with new parameters ψ_d, that is identical to the descriptor-conditioned actor $\pi_\phi(s \mid d)$, in terms of state-action mapping. In the following, we show that, given a descriptor d, we can 'specialize' the versatile descriptor-conditioned actor into a non-descriptor-conditioned policy with the same architecture as the policies stored in the archive. By sampling multiple descriptors, we can perform several actor injections and attempt to add specialized versions of the versatile actor in niches where it is high-performing, circumventing the need for expensive PG variations.

We denote the concatenation operator between two vectors by $||$, the weights and biases of the first layer of the descriptor-conditioned actor by $\mathbf{W}$ and $\mathbf{b}$. Given any states s and a descriptor d, we can compute the first layer of the descriptor-conditioned actor as $(s||d)^\intercal \mathbf{W} + \mathbf{b} = s^\intercal \mathbf{W}_1 + (d^\intercal \mathbf{W}_2 + \mathbf{b})$, with $\mathbf{W}_1$ a matrix of dimension $(\dim(\mathcal{S}), 128)$ and $\mathbf{W}_2$ a matrix of dimension $(\dim(\mathcal{D}), 128)$. Therefore, we can reinterpret the computation of the first layer as the state s multiplied with the matrix $\mathbf{W}_1$ plus the bias $d^\intercal \mathbf{W}_2 + \mathbf{b}$. Notice that the matrix $\mathbf{W}_1$ and bias $d^\intercal \mathbf{W}_2 + \mathbf{b}$ have the same dimension as the policies in the archive. Thus, if the remaining layers have the same size, we can recompute the parameters of the first layer, in order to match the architectures and inject the specialized versions of the descriptor-conditioned actor in the archive.

In DCG-MAP-Elites-AI implementation, we uniformly sample $b_{\text{AI}} = 64$ descriptors $d_1, ..., d_{b_{\text{AI}}}$ in the descriptor space $\mathcal{D}$. Then, we specialize the descriptor-conditioned actor by recomputing its parameter for each sample descriptor. At each generation, the resulting policies are suggested for addition in the archive, see Algorithm 4.

5 EXPERIMENTS

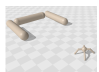
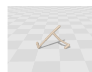
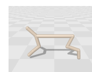
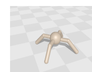
Each experiment is replicated 20 times with random seeds, over one million evaluations and the implementations are based on the QDax library [3]. The full source code will be made available upon acceptance, in a containerized environment in which all the experiments and figures can be reproduced. For the quantitative results, we report p-values based on the Wilcoxon–Mann–Whitney U test with Holm-Bonferroni correction.

5.1 Tasks

We evaluate DCG-MAP-Elites-AI on seven continuous control locomotion QD tasks [31] implemented in Brax [18] and derived from standard RL benchmarks, see Table 1. Ant Omni, AntTrap Omni and Humanoid Omni are *omnidirectional* tasks, in which the objective is to minimize energy consumption and the descriptor is the final position of the agent. Walker Uni, HalfCheetah Uni, Ant Uni and Humanoid Uni are *unidirectional* task in which the objective is to go forward as fast as possible while minimizing energy consumption and the descriptor is the feet contact rate for each foot of the agent. Walker Uni, HalfCheetah Uni, Ant Uni were introduced in PGA-MAP-Elites paper [31] and Humanoid Uni, Ant Omni, Humanoid Omni were introduced by Flageat et al. [15]. AntTrap Omni is adapted from QD-PG paper [34], the only difference being the elimination of the forward term in the reward function. We introduce AntTrap Omni to evaluate DCG-MAP-Elites-AI on a deceptive, omnidirectional environment. The trap creates a discontinuity of fitness in the descriptor space as points on both sides of the trap are close, but require two different trajectories to achieve these descriptors. Thus, the descriptor-conditioned critic needs to learn that discontinuity to provide accurate policy gradients.

PGA-MAP-Elites has previously shown state-of-the-art results on unidirectional tasks, in particular Walker Uni, HalfCheetah Uni and Ant Uni, but tends to struggle on omnidirectional tasks. In omnidirectional tasks, the global maximum of the fitness function is a solution that does not move, which is directly opposed to discovering how to reach different locations. Hence, the offsprings generated by the PG variation operator will tend to move less and travel a shorter distance. Instead, DCG-MAP-Elites-AI aims to improve the energy consumption while maintaining the ability to reach distant locations.

Table 1. Evaluation Tasks

	Ant Omni	AntTrap Omni	Humanoid Omni	Walker Uni	HalfCheetah Uni	Ant Uni	Humanoid Uni
							
State	Position and velocity of body parts						
Action	Torques applied at the hinge joints						
State dim	30	30	245	18	19	30	245
Action dim	8	8	17	6	6	8	17
Descriptor dim	2	2	2	2	2	4	2
Episode len	250	250	1000	1000	1000	1000	1000
Parameters	21,512	21,512	50,193	19,718	19,846	21,512	50,193

5.2 Main Results

5.2.1 Baselines. We compare DCG-MAP-Elites-AI with four state-of-the-art algorithms, namely MAP-Elites [43], MAP-Elites-ES [7], PGA-MAP-Elites [31] and QD-PG [34].

5.2.2 Metrics. We consider the QD score, coverage and max fitness to evaluate the final populations (i.e. archives) of all algorithms throughout training, as defined in Flageat et al. [15], Pugh et al. [35] and used in PGA-MAP-Elites paper [31]. The main metric is the *QD score*, which represents the sum of fitness of all solutions stored in the archive. This metric captures both the quality and the diversity of the population. In the tasks considered, the fitness is always positive, which avoids penalizing algorithms for finding additional solutions. We also consider the *coverage*, which represents the proportion of filled cells in the archive, measuring descriptor space illumination. Finally, we also report the *max fitness*, which is defined as the fitness of the best solution in the archive.

5.2.3 Results. The experimental results presented in Figure 2 demonstrate that DCG-MAP-Elites-AI achieves equal or higher QD score and coverage than all baselines on all tasks, especially PGA-MAP-Elites, the previous state-of-the-art. On Ant Uni and Humanoid Uni, DCG-MAP-Elites-AI achieves a higher median QD score but not significantly. On all other tasks, DCG-MAP-Elites-AI achieves a significantly higher QD score ($p < 0.003$), demonstrating that our method generates populations of solutions that are higher-performing and more diverse. Especially, the coverage metric shows that DCG-MAP-Elites-AI surpasses the exploration capabilities of QD-PG on all tasks ($p < 0.05$). DCG-MAP-Elites-AI significantly outperforms the GECCO version [12] on all environments except Ant Uni ($p < 0.01$), where they perform similarly, showing that the improvements made to the algorithm are beneficial. DCG-MAP-Elites-AI also achieves equal or significantly better max fitness on all environments except on HalfCheetah Uni and Ant Uni, where PGA-MAP-Elites is better, showing room for improvement. Finally, we also show that our method still benefits from the exploration power of the GA operator even in deceptive environment like AntTrap Omni. The experimental results confirm that DCG-MAP-Elites-AI is able to overcome the limits of PGA-MAP-Elites on omnidirectional tasks while performing better on the unidirectional tasks ($p < 0.005$) except Ant Uni where our method is not significantly better. Thus, confirming the interest of having a descriptor-conditioned gradient to make the PG variation operator fruitful in a wider range of tasks. Overall, DCG-MAP-Elites-AI shows competitive performance on all metrics and tasks, hence proving to be the first successful effort in the QD-RL literature to achieve well on both the unidirectional and omnidirectional tasks. Previous efforts were usually adapted to either one or the other [31, 34, 42].

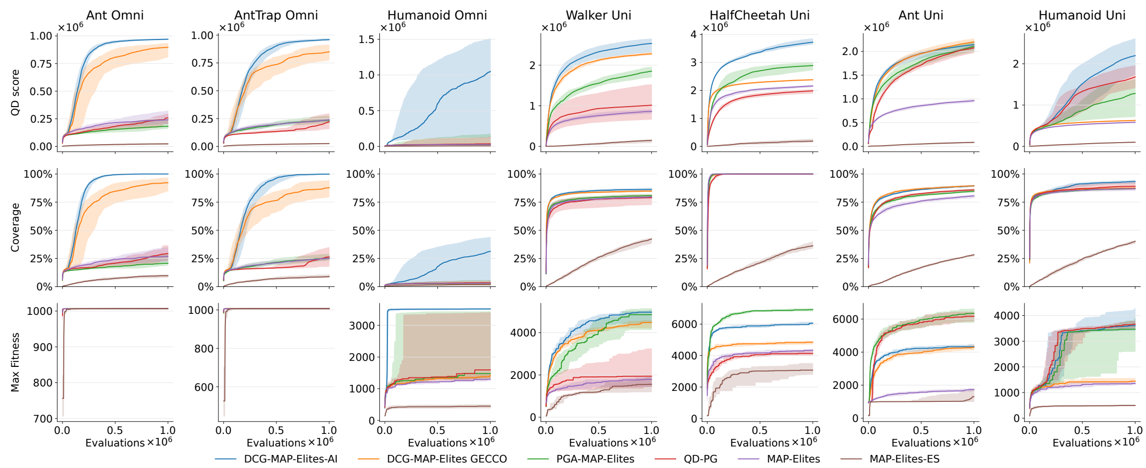


Fig. 2. QD score, coverage and max fitness (Section 5.2.2) for DCG-MAP-Elites-AI and all baselines on all tasks. Each experiment is replicated 20 times with random seeds. The solid line is the median and the shaded area represents the first and third quartiles.

Qualitative results in Figure 3 also show that DCG-MAP-Elites-AI discovers solutions that are more diverse and higher-performing than other baselines on Ant Omni task. The final archives for all algorithms and on all tasks are provided in Appendix A.1.

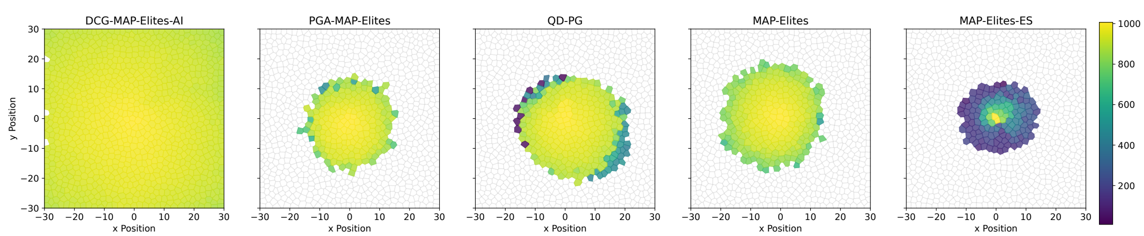


Fig. 3. **Ant Omni** Archive at the end of training for all algorithms.

5.3 Ablations

5.3.1 Ablation studies. We also compare DCG-MAP-Elites-AI with three ablations, namely DCG-MAP-Elites GECCO [12], Ablation AI and Ablation Actor. In DCG-MAP-Elites GECCO, there is no actor injection, but we perform actor evaluation instead to provide on-policy samples to the TD3 algorithm. In Ablation AI, there is no actor injection and no actor evaluation. In Ablation Actor, the actor is not descriptor-conditioned, removing the archive distillation component, but the critic is still descriptor-conditioned.

5.3.2 Results. We perform two ablation experiments to show the importance of actor injection and of the descriptor-conditioned actor. AI proves significantly beneficial in terms of QD score, on all tasks ($p < 0.05$) except Ant Uni where they perform comparably. Having a descriptor-conditioned actor $\pi_\phi(\,.\mid d)$ rather than a normal actor $\pi_\phi(\,.\,)$ proves significantly beneficial in terms of QD score, on all tasks ($p < 10^{-4}$), demonstrating that the descriptor-conditioned actor enables archive distillation while being beneficial for the critic's training. DCG-MAP-Elites GECCO achieves equal or higher QD score than the AI ablation, showing the importance of on-policy samples. Overall, DCG-MAP-Elites-AI shows competitive performance on all metrics and tasks compared to the ablations, hence proving the importance of the different enhancements compared to PGA-MAP-Elites.

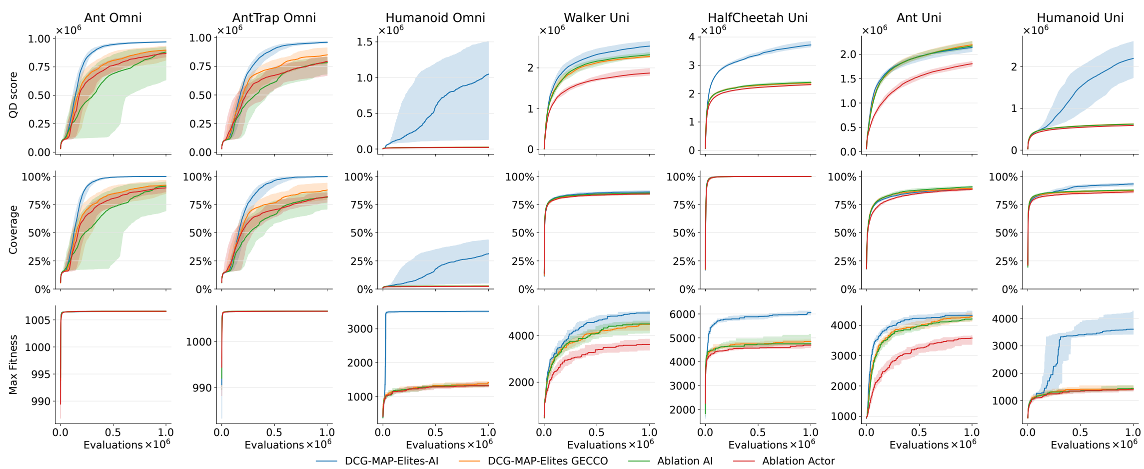


Fig. 4. QD score, coverage and max fitness (Section 5.2.2) for DCG-MAP-Elites-AI and the ablations on all tasks. Each experiment is replicated 20 times with random seeds. The solid line is the median and the shaded area represents the first and third quartiles.

5.4 Reproducibility

5.4.1 Reproducibility Metrics. We also consider three metrics to evaluate the reproducibility of the final archives for all algorithms and of the descriptor-conditioned actor for DCG-MAP-Elites-AI, at the end of training. QD algorithms based on MAP-Elites output a population of solutions that we evaluate with the QD score, coverage and max fitness, see Section 5.2.2. However, these metrics can be misleading because in stochastic environments, a solution might give different fitnesses and descriptors when evaluated multiple times. Consequently, the QD score, coverage and max fitness can be overestimated, an effect that is well-known and that has been studied in the past [14]. An archive of solutions is considered reproducible, if the QD score, coverage and max fitness does not change substantially after multiple reevaluation of the individuals. Thus, to assess the reproducbility of the archives, we consider the *expected*

QD score, the *expected distance to descriptor* and the *expected max fitness*. To calculate those metrics, we reevaluate each solution in the archive 512 times, to approximate its expected fitness and expected distance to descriptor. The expected distance to descriptor of a solution is simply the expected euclidean distance between the descriptor of the cell of the individual and the observed descriptors. Therefore, for the expected distance to descriptor, lower is better. We use the expected fitness and expected distance to descriptor of all solutions to calculate the expected QD score, expected distance to descriptor and expected max fitness of the archive.

Additionally, DCG-MAP-Elites-AI's descriptor-conditioned actor can in principle achieve different descriptors and thus, is comparable to an archive. Similarly to the archive, we evaluate its expected QD score, expected distance to descriptor and expected max fitness. To that end, we take the descriptor d of each filled cell in the corresponding archive, and evaluate the actor $\pi_\phi(\,.\mid d)$ 512 times, to approximate its expected fitness and expected distance to descriptor. Analogously to the archive, we use the expected fitnesses and expected distances to descriptor to calculate the expected QD score, expected distance to descriptor and expected max fitness of the descriptor-conditioned actor.

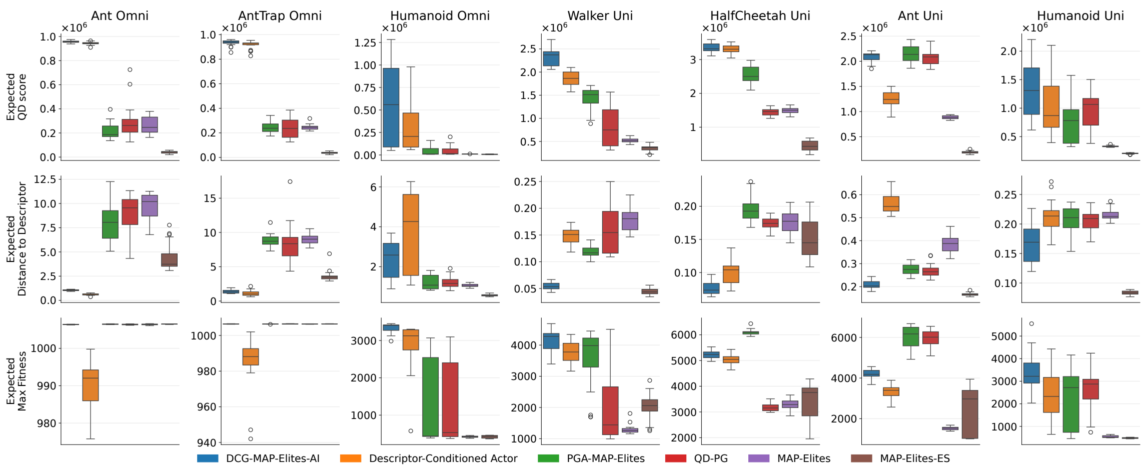


Fig. 5. Expected QD score, expected distance to descriptor (lower is better) and expected max fitness (Section 5.4.1) for DCG-MAP-Elites-AI, the descriptor-conditioned policy and the baselines on all tasks. Each experiment is replicated 20 times with random seeds.

5.4.2 Results. In Figure 5, we provide the expected QD score, expected distance to descriptor and expected max fitness of the final archive and the descriptor-conditioned policy, see Section 5.4.1. First, we can see that DCG-MAP-Elites-AI's final archive achieves equal or higher expected QD score than all baselines on all tasks. The descriptor-conditioned actor performs similarly to DCG-MAP-Elites-AI on most environments, but performs significantly worse on Ant Uni. This shows that, in most cases, the descriptor-conditioned actor is able to restore the quality of the archive although having compressed the information in a single network. Second, DCG-MAP-Elites-AI obtains better expected distance to descriptor (lower is better) than all baselines except MAP-Elites-ES on all tasks. However, MAP-Elites-ES obtains worse QD score and most importantly, worst coverage, making it easier for MAP-Elites-ES to achieve a low expected distance to descriptor. DCG-MAP-Elites-AI descriptor-conditioned actor obtains similar expected distance to descriptor on omnidirectional. However, it performs consistently worse on unidirectional tasks. This shows that in some cases, while compressing the quality of the archive in a single network, the descriptor-conditioned actor can also exhibit the same diversity as the population. Those two combined observations show that the final archive and descriptor-conditioned policy have similar properties on omnidirectional tasks. Overall, those results show that our single descriptor-conditioned policy can already be seen as a promising summary of our archive, showing very similar properties on half our tasks.

5.5 Variation Operators Evaluation

5.5.1 Variation Operator Metrics. DCG-MAP-Elites-AI and PGA-MAP-Elites make use of a GA variation operator and of a PG variation operator. The GA variation operator is strictly the same in both algorithms. However, DCG-MAP-Elites-AI enhances PGA-MAP-Elites's PG variation operator with a descriptor-conditioned critic, as explained in Section 4.4. To evaluate the performance of each variation operator, we introduce a metric defined as the accumulated number of offsprings added to the archive coming from each variation operator throughout training, that we call *number of elites*. By tracking the number of elites generated by each variation operator over the course of training, we can analyze the interaction and dynamics between the different variation operators and actor injection, providing insights into the relative contributions of the different components.

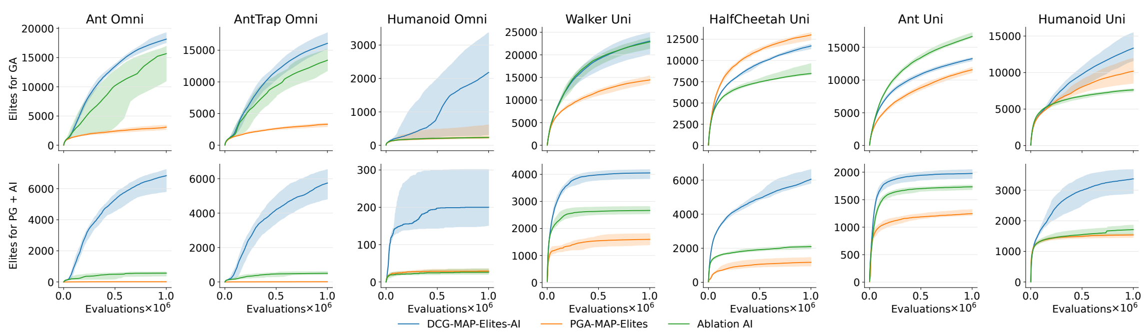


Fig. 6. Accumulated number of offsprings added to the archive (Section 5.5.1) for (**top**) GA variation operator and (**bottom**) PG variation operator plus Actor Injection (AI). Each experiment is replicated 20 times with random seeds. The solid line is the median and the shaded area represents the first and third quartiles.

5.5.2 Results. On the top row of Figure 6, we can see the accumulated number of elites for the GA variation operator for DCG-MAP-Elites-AI, PGA-MAP-Elites and ablation AI throughout training. In all three cases, the number of offsprings suggested for addition in the archive is 128. On the bottom row of Figure 6, we can see the accumulated number of elites for the PG variation operator. In all three cases, the number of offsprings suggested for addition in the archive is 128, but for DCG-MAP-Elites-AI, the PG variation is divided into 64 coming from the actor injection (Section 4.5) and 64 coming from the PG update using the descriptor-conditioned critic (Section 4.4). First, we can see that the ablation of the actor injection generates a larger number of elites than PGA-MAP-Elites, demonstrating that the descriptor-conditioned critic generates higher-performing and more diverse solution than the traditional critic used in PGA-MAP-Elites. Furthermore, we can see that DCG-MAP-Elites-AI with actor injection mechanism generates even more elites than the descriptor-conditioned PG variation operator alone. Interestingly, we can see that the number of elites generated by DCG-MAP-Elites-AI is higher than PGA-MAP-Elites, even though the GA variation operators are exactly the same. This demonstrates that the solutions found by the descriptor-conditioned PG variation operator are better stepping stones.

6 CONCLUSION

In this work, we introduce DCG-MAP-Elites-AI and demonstrate the benefits of having descriptor-conditioned gradients to evolve populations of large neural networks. We concurrently train a descriptor-conditioned actor, as a by-product of the critic's training, that can achieve a diversity of high-performing behaviors. In turn, we inject the trained descriptor-conditioned actor in the population, despite network architecture differences, speeding-up training even more. Our method, DCG-MAP-Elites-AI, achieves equal or better performance than all baselines on seven continuous control locomotion tasks. We also show that the synergy between the fitness improvement capabilities of the PG variations and the exploration capabilities of the GA variations is preserved, even in deceptive environments. The descriptor-conditioned actor demonstrates performance that are similar to the discrete archive, summarizing its capabilities into one single neural network and acting as a continuous archive. We think that distilling the archive into a single policy is a promising method as it enables to have less redundancy compared to a discrete archive in which most of the solutions can be similar, especially between close cells. The descriptor-conditioned policy can also negate the burden of dealing with an archive of thousands of solutions in practical applications.

The benefits of combining RL methods with PGA-MAP-Elites come with the limitations of MDP settings. Specifically, we are limited to evolving differentiable solutions and the foundations of RL algorithms rely on the Markov property and full observability. In this work in particular, we face challenges with the Markov property because the descriptors depend on full trajectories. Thus, the scaled reward introduced in our method depends on the full trajectory and not only on the current state and action. The performance of the descriptor-conditioned policy also shows that there is room for improvement to better distill the knowledge of the archive.

For future work, we would like to investigate the generalization capabilities of the descriptor-conditioned policy trained with DCG-MAP-Elites-AI and try to produce solutions with descriptors that are not in the archive, performing descriptor space generalization. In our method, the critic attempts to mutate solutions to produce offspring with higher fitness while keeping their descriptors constant. We think that we could use the descriptor-conditioned critic to mutate solutions to produce offspring towards different descriptors, thereby explicitly promoting diversity.

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html

[2] Felix Chalumeau, Raphael Boige, Bryan Lim, Valentin Macé, Maxime Allard, Arthur Flajolet, Antoine Cully, and Thomas Pierrot. 2022. Neuroevolution is a Competitive Alternative to Reinforcement Learning for Skill Discovery. https://openreview.net/forum?id=6BHlZgyPOZY

[3] Felix Chalumeau, Bryan Lim, Raphael Boige, Maxime Allard, Luca Grillotti, Manon Flageat, Valentin Macé, Arthur Flajolet, Thomas Pierrot, and Antoine Cully. 2023. QDax: A Library for Quality-Diversity and Population-based Algorithms with Hardware Acceleration. arXiv:2308.03665 [cs.AI]

[4] Felix Chalumeau, Thomas Pierrot, Valentin Macé, Arthur Flajolet, Karim Beguir, Antoine Cully, and Nicolas Perrin-Gilbert. 2023. Assessing Quality-Diversity Neuro-Evolution Algorithms Performance in Hard Exploration Problems. https://doi.org/10.48550/arXiv.2211.13742 arXiv:2211.13742 [cs].

[5] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2021. Quality-Diversity Optimization: A Novel Branch of Stochastic Optimization. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, Panos M. Pardalos, Varvara Rasskazova, and Michael N. Vrahatis (Eds.). Springer International Publishing, Cham, 109–135. https://doi.org/10.1007/978-3-030-66515-9_4

[6] Konstantinos Chatzilygeroudis, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2018. Reset-free Trial-and-Error Learning for Robot Damage Recovery. *Robotics and Autonomous Systems* 100 (Feb. 2018), 236–250. https://doi.org/10.1016/j.robot.2017.11.010

[7] Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. 2020. Scaling MAP-Elites to deep neuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 67–75. https://doi.org/10.1145/3377930.3390217

[8] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (May 2015), 503–507. https://doi.org/10.1038/nature14422 Number: 7553 Publisher: Nature Publishing Group.

[9] Antoine Cully and Yiannis Demiris. 2018. Quality and Diversity Optimization: A Unifying Modular Framework. *IEEE Transactions on Evolutionary Computation* 22, 2 (2018), 245–259. https://doi.org/10.1109/TEVC.2017.2704781

[10] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2021. First return, then explore. *Nature* 590, 7847 (Feb. 2021), 580–586. https://doi.org/10.1038/s41586-020-03157-9 Number: 7847 Publisher: Nature Publishing Group.

[11] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2018. Diversity is All You Need: Learning Skills without a Reward Function. https://doi.org/10.48550/arXiv.1802.06070 arXiv:1802.06070 [cs].

[12] Maxence Faldor, Félix Chalumeau, Manon Flageat, and Antoine Cully. 2023. MAP-Elites with Descriptor-Conditioned Gradients and Archive Distillation into a Single Policy. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 138–146. https://doi.org/10.1145/3583131.3590503

[13] Manon Flageat, Félix Chalumeau, and Antoine Cully. 2023. Empirical analysis of PGA-MAP-Elites for Neuroevolution in Uncertain Domains. *ACM Transactions on Evolutionary Learning and Optimization* 3, 1 (March 2023), 1:1–1:32. https://doi.org/10.1145/3577203

[14] Manon Flageat and Antoine Cully. 2023. Uncertain Quality-Diversity: Evaluation methodology and new methods for Quality-Diversity in Uncertain Domains. *IEEE Transactions on Evolutionary Computation* (2023).

[15] Manon Flageat, Bryan Lim, Luca Grillotti, Maxime Allard, Simón C. Smith, and Antoine Cully. 2022. Benchmarking Quality-Diversity Algorithms on Neuroevolution for Reinforcement Learning. https://doi.org/10.48550/arXiv.2211.02193 arXiv:2211.02193 [cs].

[16] Matthew Fontaine and Stefanos Nikolaidis. 2021. Differentiable Quality Diversity. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 10040–10052. https://proceedings.neurips.cc/paper/2021/hash/532923f11ac97d3e7cb0130315b067dc-Abstract.html

[17] Matthew Fontaine and Stefanos Nikolaidis. 2023. Covariance Matrix Adaptation MAP-Annealing. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 456–465. https://doi.org/10.1145/3583131.3590389

[18] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. 2021. *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation.* http://github.com/google/brax

[19] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 1587–1596. https://proceedings.mlr.press/v80/fujimoto18a.html ISSN: 2640-3498.

[20] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. 2016. Variational Intrinsic Control. https://doi.org/10.48550/arXiv.1611.07507 arXiv:1611.07507 [cs].

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 1861–1870. https://proceedings.mlr.press/v80/haarnoja18b.html ISSN: 2640-3498.

[22] Nikolaus Hansen. 2023. The CMA Evolution Strategy: A Tutorial. https://doi.org/10.48550/arXiv.1604.00772 arXiv:1604.00772 [cs, stat].

[23] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, and David Silver. 2017. Emergence of Locomotion Behaviours in Rich Environments. (July 2017).

[24] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

[25] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Population Based Training of Neural Networks. https://doi.org/10.48550/arXiv.1711.09846 arXiv:1711.09846 [cs].

[26] Marija Jegorova, Stéphane Doncieux, and Timothy Hospedales. 2019. Behavioural Repertoire via Generative Adversarial Policy Networks. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. https://doi.org/10.1109/ICDL-EpiRob44920.2019 arXiv:1811.02945 [cs, stat].

[27] Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. 2020. One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 8198–8210. https://proceedings.neurips.cc/paper/2020/hash/5d151d1059a6281335a10732fc49620e-Abstract.html

[28] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1509.02971

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533. https://doi.org/10.1038/nature14236 Number: 7540 Publisher: Nature Publishing Group.

[30] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *CoRR* abs/1504.04909 (2015). arXiv:1504.04909 http://arxiv.org/abs/1504.04909

[31] Olle Nilsson and Antoine Cully. 2021. Policy gradient assisted MAP-Elites. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 866–875. https://doi.org/10.1145/3449639.3459304

[32] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. 2019. Solving Rubik's Cube with a Robot Hand. https://doi.org/10.48550/arXiv.1910.07113 arXiv:1910.07113 [cs, stat].

[33] Thomas Pierrot and Arthur Flajolet. 2023. Evolving Populations of Diverse RL Agents with MAP-Elites. https://doi.org/10.48550/arXiv.2303.12803 arXiv:2303.12803 [cs].

[34] Thomas Pierrot, Valentin Macé, Felix Chalumeau, Arthur Flajolet, Geoffrey Cideron, Karim Beguir, Antoine Cully, Olivier Sigaud, and Nicolas Perrin-Gilbert. 2022. Diversity policy gradient for sample efficient quality-diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 1075–1083. https://doi.org/10.1145/3512290.3528845

[35] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI* 3 (2016). https://www.frontiersin.org/articles/10.3389/frobt.2016.00040

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

[37] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal Value Function Approximators. In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 1312–1320. https://proceedings.mlr.press/v37/schaul15.html ISSN: 1938-7228.

[38] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. 2019. Dynamics-Aware Unsupervised Discovery of Skills. https://openreview.net/forum?id=HJgLZR4KvH

[39] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (Jan. 2016), 484–489. https://doi.org/10.1038/nature16961 Number: 7587 Publisher: Nature Publishing Group.

[40] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning*. PMLR, 387–395. https://proceedings.mlr.press/v32/silver14.html ISSN: 1938-7228.

[41] Bryon Tjanaka, Matthew C. Fontaine, David H. Lee, Aniruddha Kalkar, and Stefanos Nikolaidis. 2023. Training Diverse High-Dimensional Controllers by Scaling Covariance Matrix Adaptation MAP-Annealing. https://doi.org/10.48550/arXiv.2210.02622 arXiv:2210.02622 [cs].

[42] Bryon Tjanaka, Matthew C. Fontaine, Julian Togelius, and Stefanos Nikolaidis. 2022. Approximating gradients for differentiable quality diversity in reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 1102–1111. https://doi.org/10.1145/3512290.3528705

[43] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2018. Using Centroidal Voronoi Tessellations to Scale Up the Multidimensional Archive of Phenotypic Elites Algorithm. *IEEE Transactions on Evolutionary Computation* 22, 4 (2018), 623–630. https://doi.org/10.1109/TEVC.2017.2735550

[44] Vassiiis Vassiliades and Jean-Baptiste Mouret. 2018. Discovering the elite hypervolume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 149–156. https://doi.org/10.1145/3205455.3205602

[45] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (Nov. 2019), 350–354. https://doi.org/10.1038/s41586-019-1724-z Number: 7782 Publisher: Nature Publishing Group.

A SUPPLEMENTARY RESULTS

A.1 Archives

We provide the archives obtained at the end of training for each algorithm on all environments. For each (algorithm, environment) pair, we select the most representative seed with the QD score closest to the median QD score over all seeds to avoid cherry picking.

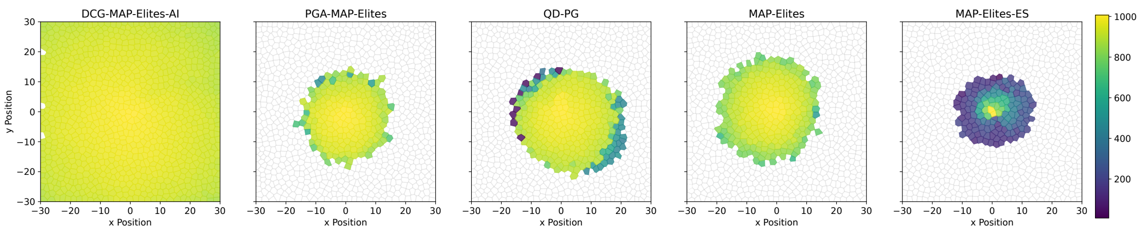


Fig. 7. **Ant Omni** Archive at the end of training for all algorithms.

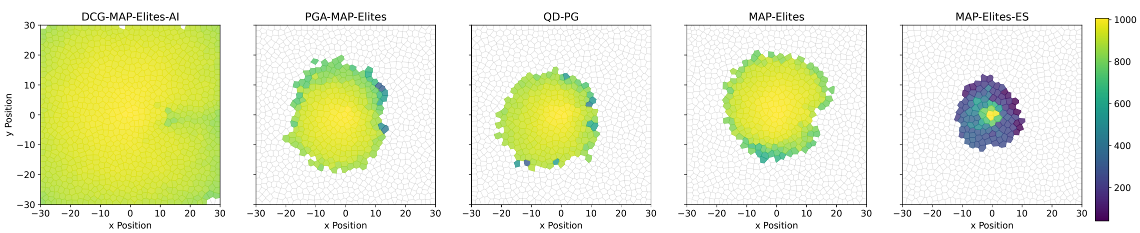


Fig. 8. **AntTrap Omni** Archive at the end of training for all algorithms.

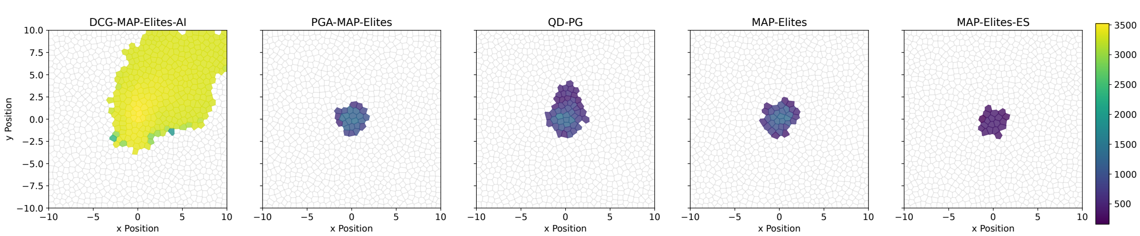


Fig. 9. **Humanoid Omni** Archive at the end of training for all algorithms.

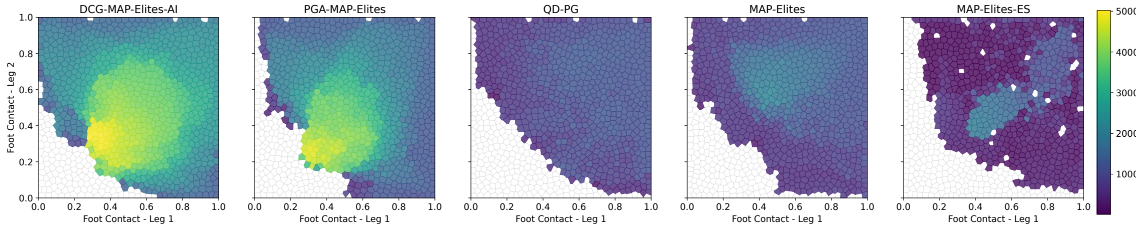


Fig. 10. **Walker Uni** Archive at the end of training for all algorithms.

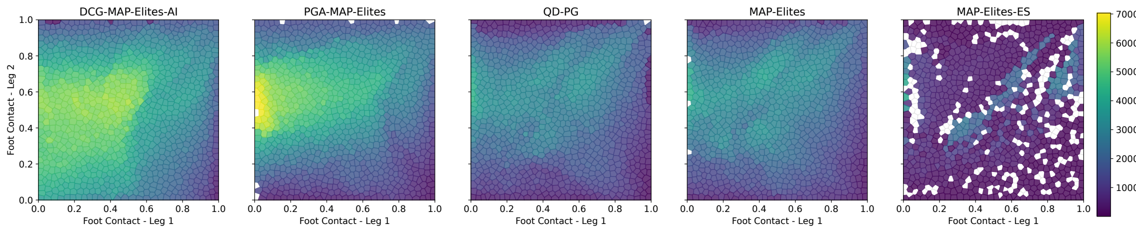


Fig. 11. **Halfcheetah Uni** Archive at the end of training for all algorithms.

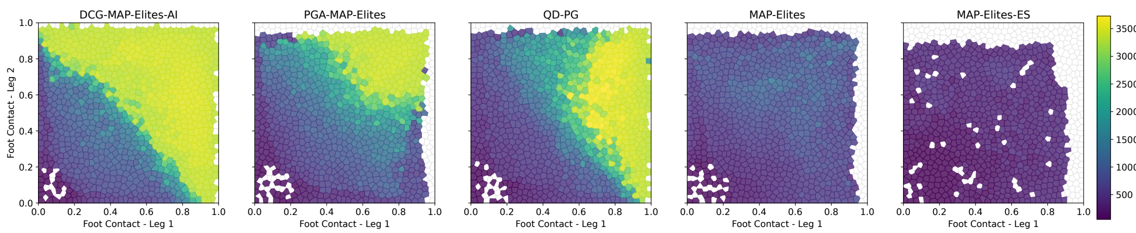


Fig. 12. **Humanoid Uni** Archive at the end of training for all algorithms.

B ALGORITHMS

B.1 DCG-MAP-Elites-AI

Algorithm 5 DCG-MAP-Elites-AI

Require: GA batch size b_{GA}, PG batch size b_{PG}, Actor Injection batch size b_{AI}, total batch size $b = b_{\text{GA}} + b_{\text{PG}} + b_{\text{AI}}$
  Initialize archive $\mathcal{X}$ with b random solutions and replay buffer $\mathcal{B}$
  Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_ϕ
  $i \leftarrow 0$
  while $i < I$ **do**
    $\textsc{train\_actor\_critic}(\pi_\phi, Q_{\theta_1}, Q_{\theta_2}, \mathcal{B})$
    $\pi_{\psi_1}, \dots, \pi_{\psi_b} \leftarrow \textsc{selection}(\mathcal{X})$
    $\pi_{\widehat{\psi}_1}, \dots, \pi_{\widehat{\psi}_{b_{\text{GA}}}} \leftarrow \textsc{variation\_ga}(\pi_{\psi_1}, \dots, \pi_{\psi_{b_{\text{GA}}}})$
    $\pi_{\widehat{\psi}_{b_{\text{GA}}+1}}, \dots, \pi_{\widehat{\psi}_{b_{\text{GA}}+b_{\text{PG}}}} \leftarrow \textsc{variation\_pg}(\pi_{\psi_{b_{\text{GA}}+1}}, \dots, \pi_{\psi_{b_{\text{GA}}+b_{\text{PG}}}}, Q_{\theta_1}, \mathcal{B})$
    $\pi_{\widehat{\psi}_{b_{\text{GA}}+b_{\text{PG}}+1}}, \dots, \pi_{\widehat{\psi}_b} \leftarrow \textsc{actor\_injection}(\pi_\phi)$
    $\textsc{addition}(\pi_{\widehat{\psi}_1}, \dots, \pi_{\widehat{\psi}_b}, \mathcal{X}, \mathcal{B})$
    $i \leftarrow i + b$
  function $\textsc{addition}(\pi_{\widehat{\psi}} \dots, \mathcal{X}, \mathcal{B})$
    for $\pi_{\widehat{\psi}} \dots$ **do**
      $(f, \text{transitions}) \leftarrow F(\pi_{\widehat{\psi}}),\ d \leftarrow D(\pi_{\widehat{\psi}})$
      $\textsc{insert}(\mathcal{B}, \text{transitions})$
      if $\mathcal{X}(d) = \emptyset$ or $F(\mathcal{X}(d)) < f$ **then**
        $\mathcal{X}(d) \leftarrow \pi_{\widehat{\psi}}$

Algorithm 6 Descriptor-conditioned Actor-Critic Training

  function $\textsc{train\_actor\_critic}(\pi_\phi, Q_{\theta_1}, Q_{\theta_2}, \mathcal{B})$
    for $t = 1 \to n$ **do**
      Sample N transitions (s, a, r, s', d, d') from $\mathcal{B}$
      Sample smoothing noise ϵ
      $y \leftarrow S(d, d')\, r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi'}(s' \mid d') + \epsilon \mid d')$
      Update both critics by regression to y
      if t mod Δ **then**
        Update actor using the deterministic policy gradient:
        $\frac{1}{N} \sum \nabla_\phi \pi_\phi(s \mid d') \nabla_a Q_{\theta_1}(s, a \mid d')|_{a=\pi_\phi(s|d')}$
        Soft-update target networks $Q_{\theta i'}$ and $\pi_{\phi'}$

Algorithm 7 Descriptor-conditioned PG Variation

function VARIATION_PG($\pi_{\psi} \ldots, Q_{\theta_1}, \mathcal{B}$)
  for $\pi_{\psi} \ldots$ **do**
    $d_{\psi} \leftarrow D(\pi_{\psi})$
    for $i = 1 \longrightarrow m$ **do**
      Sample N transitions (s, a, r, s', d, d') from $\mathcal{B}$
      Update actor using the deterministic policy gradient:
      $\frac{1}{N} \sum \nabla_{\psi} \pi_{\psi}(s) \nabla_a Q_{\theta_1}(s, a \mid d_{\psi})|_{a=\pi_{\psi}(s)}$
  return $\pi_{\widehat{\phi}} \ldots$

Algorithm 8 Descriptor-conditioned Actor Injection

function ACTOR_INJECTION(π_{ϕ})
  $d_1, \ldots, d_{b_{\text{AI}}} \sim \mathcal{U}(\mathcal{D})$
  $\psi_1, \ldots, \psi_{b_{\text{AI}}} \leftarrow$ PARAMETERS_RECOMPUTATION($\pi_{\phi}(. \mid d_1), \ldots, \pi_{\phi}(. \mid d_{b_{\text{AI}}})$)
  return $\pi_{\psi_1}, \ldots, \pi_{\psi_{b_{\text{AI}}}}$

B.2 PGA-MAP-Elites

Algorithm 9 PGA-MAP-Elites

Require: GA batch size b_{GA}, PG batch size b_{PG}, total batch size $b = b_{\text{GA}} + b_{\text{PG}}$
  Initialize archive $\mathcal{X}$ with b random solutions and replay buffer $\mathcal{B}$
  Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_{ϕ}
  $i \leftarrow 0$
  while $i < I$ **do**
    TRAIN_ACTOR_CRITIC($\pi_{\phi}, Q_{\theta_1}, Q_{\theta_2}, \mathcal{B}$)
    $\pi_{\psi_1}, \ldots, \pi_{\psi_{b-1}} \leftarrow$ SELECTION($\mathcal{X}$)
    $\pi_{\widehat{\psi}_1}, \ldots, \pi_{\widehat{\psi}_{b_{\text{GA}}}} \leftarrow$ VARIATION_GA($\pi_{\psi_1}, \ldots, \pi_{\psi_{b_{\text{GA}}}}$)
    $\pi_{\widehat{\psi}_{b_{\text{GA}}+1}}, \ldots, \pi_{\widehat{\psi}_{b-1}} \leftarrow$ VARIATION_PG($\pi_{\psi_{b_{\text{GA}}+1}}, \ldots, \pi_{\psi_{b-1}}, Q_{\theta_1}, \mathcal{B}$)
    $\pi_{\widehat{\psi}_b} \leftarrow$ ACTOR_INJECTION(π_{ϕ})
    ADDITION($\mathcal{X}, \pi_{\widehat{\psi}_1}, \ldots, \pi_{\widehat{\psi}_{b-1}}, \pi_{\phi}, \mathcal{B}$)
    $i \leftarrow i + b$
  function ADDITION($\mathcal{X}, \pi_{\widehat{\psi}} \ldots, \mathcal{B}$)
    for $\pi_{\widehat{\psi}} \ldots$ **do**
      $(f$, transitions$) \leftarrow F(\pi_{\widehat{\psi}})$, $d \leftarrow D(\pi_{\widehat{\psi}})$
      INSERT($\mathcal{B}$, transitions)
      if $\mathcal{X}(d) = \emptyset$ or $F(\mathcal{X}(d)) < f$ **then**
        $\mathcal{X}(d) \leftarrow \pi_{\widehat{\psi}}$

Algorithm 10 Actor-Critic Training

function TRAIN_ACTOR_CRITIC($\pi_\phi, Q_{\theta_1}, Q_{\theta_2}, \mathcal{B}$)
  for $t = 1 \rightarrow n$ **do**
    Sample N transitions (s, a, r, s') from $\mathcal{B}$
    Sample smoothing noise ϵ
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi'}(s') + \epsilon)$
    Update both critics by regression to y
    if t mod Δ **then**
      Update actor using the deterministic policy gradient:
      $\frac{1}{N} \sum \nabla_\phi \pi_\phi(s) \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)}$
      Soft-update target networks $Q_{\theta i'}$ and $\pi_{\phi'}$

Algorithm 11 PG Variation

function VARIATION_PG($\pi_\psi \dots, Q_{\theta_1}, \mathcal{B}$)
  for $\pi_\psi \dots$ **do**
    for $i = 1 \rightarrow m$ **do**
      Sample N transitions (s, a, r, s') from $\mathcal{B}$
      Update actor using the deterministic policy gradient:
      $\frac{1}{N} \sum \nabla_\psi \pi_\psi(s) \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\psi(s)}$
  return $\pi_{\widehat{\psi}} \dots$

Algorithm 12 Actor Injection

function ACTOR_INJECTION(π_ϕ)
  return π_ϕ

B.3 QD-PG

Algorithm 13 QD-PG

Require: GA batch size b_{GA}, QPG batch size b_{QPG}, DPG batch size b_{DPG}, total batch size $b = b_{\mathrm{GA}} + b_{\mathrm{QPG}} + b_{\mathrm{DPG}}$

```
Initialize archive 𝒳 with b random solutions and replay buffer ℬ
Initialize critic networks Q_{θ_Q}, Q_{θ_D} and actor network π_φ
i ← 0
while i < I do
    TRAIN_ACTOR_CRITIC(π_φ, Q_{θ_Q}, Q_{θ_D}, ℬ)
    π_{ψ_1}, ..., π_{ψ_b} ← SELECTION(𝒳)
    π_{ψ̂_1}, ..., π_{ψ̂_{b_GA}} ← VARIATION_GA(π_{ψ_1}, ..., π_{ψ_{b_GA}})
    π_{ψ̂_{b_GA+1}}, ..., π_{ψ̂_{b_GA+b_QPG}} ← VARIATION_QPG(π_{ψ_{b_GA+1}}, ..., π_{ψ_{b_GA+b_QPG}}, Q_{θ_Q}, ℬ)
    π_{ψ̂_{b_GA+b_QPG+1}}, ..., π_{ψ̂_b} ← VARIATION_DPG(π_{ψ_{b_GA+b_QPG+1}}, ..., π_{ψ_b}, Q_{θ_D}, ℬ)
    ADDITION(π_{ψ̂_1}, ..., π_{ψ̂_b}, 𝒳, ℬ)
    i ← i + b
function ADDITION(𝒳, ℬ, π_φ, π_ψ̂ ...)
    for d' ∈ 𝒟 sampled from b solutions in 𝒳 do
        (f, transitions) ← F(π_φ( . | d'))
        INSERT(ℬ, transitions)
    for π_ψ̂ ... do
        (f, transitions) ← F(π_ψ̂), d ← D(π_ψ̂)
        INSERT(ℬ, transitions)
        if 𝒳(d) = ∅ or F(𝒳(d)) < f then
            𝒳(d) ← π_ψ̂
```

B.4 MAP-Elites

Algorithm 14 MAP-Elites

Require: GA batch size b_{GA}

```
Initialize archive 𝒳 with b_GA random solutions
i ← 0
while i < I do
    x_1, ..., x_{b_GA} ← SELECTION(𝒳)
    x̂_1, ..., x̂_{b_GA} ← VARIATION(x_1, ..., x_{b_GA})
    ADDITION(𝒳, x̂_1, ..., x̂_{b_GA})
    i ← i + b_GA
function ADDITION(𝒳, x̂ ...) :
    for x̂ ... do
        f ← F(x̂), d ← D(x̂)
        if 𝒳(d) = ∅ or F(𝒳(d)) < f then
            𝒳(d) ← x̂
```

B.5 MAP-Elites-ES

Algorithm 15 MAP-Elites-ES

Require: Number of ES samples N, standard deviation of ES samples σ, explore-exploit alternation N_{gen}, number of re-sampling M

Initialize archive $\mathcal{X}$ with N random solutions, initialise empty novelty archive $\mathcal{A}$

$i \leftarrow 0$

while $i < I$ **do**

  if $i\%N_{gen} == 0$ **then**:

    $x \leftarrow \text{SELECTION\_EXPLOIT}(\mathcal{X})$

    $\widehat{x} \leftarrow \text{VARIATION\_EXPLOIT}(x)$

  else:

    $x \leftarrow \text{SELECTION\_EXPLORE}(\mathcal{X})$

    $\widehat{x} \leftarrow \text{VARIATION\_EXPLORE}(\mathcal{A}, x)$

  $\text{ADDITION}(\mathcal{X}, \mathcal{A}, \widehat{x})$

  $i \leftarrow i + N + M$

function $\text{ADDITION}(\mathcal{X}, \mathcal{A}, \widehat{x})$:

  for $i = 1, \ldots, M$ **do**

    $f_i \leftarrow F(\widehat{x})), d_i \leftarrow D(\widehat{x}))$

  $f \leftarrow \text{average}(f_i)$, $d \leftarrow \text{average}(d_i)$

  $\mathcal{A} \leftarrow \mathcal{A} + d$

  if $\mathcal{X}(d) = \emptyset$ or $F(\mathcal{X}(d)) < f$ **then**

    $\mathcal{X}(d) \leftarrow \widehat{x}$

function $\text{VARIATION\_EXPLOIT}(x)$:

  $x_1, \ldots, x_N \leftarrow \text{SAMPLE\_GAUSSIAN}(x, \sigma)$

  $f_1, \ldots, f_N \leftarrow F(x_1, \ldots, x_N)$

  $\widehat{x} \leftarrow \text{ES\_STEP}(x, f_1, \ldots, f_N)$

function $\text{VARIATION\_EXPLORE}(\mathcal{A}, x)$:

  $x_1, \ldots, x_N \leftarrow \text{SAMPLE\_GAUSSIAN}(x, \sigma)$

  $d_1, \ldots, d_N \leftarrow D(x_1, \ldots, x_N)$

  $nov_1, \ldots, nov_N \leftarrow \text{NOVELTY}(\mathcal{A}, d_1, \ldots, d_N)$

  $\widehat{x} \leftarrow \text{ES\_STEP}(x, nov_1, \ldots, nov_N)$

C HYPERPARAMETERS

C.1 DCG-MAP-Elites-AI

Table 2. DCG-MAP-Elites-AI hyperparameters

Parameter	Value
Number of centroids	1024
Total batch size b	256
GA batch size b_{GA}	128
PG batch size b_{PG}	64
AI batch size b_{AI}	64
Policy networks	$[128, 128, \lvert\mathcal{A}\rvert]$
GA variation param. 1 σ_1	0.005
GA variation param. 2 σ_2	0.05
Actor network	$[128, 128, \lvert\mathcal{A}\rvert]$
Critic network	$[256, 256, 1]$
TD3 batch size N	100
Critic training steps n	3000
PG training steps m	150
Policy learning rate	5×10^{-3}
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Replay buffer size	10^6
Discount factor γ	0.99
Actor delay Δ	2
Target update rate	0.005
Smoothing noise var. σ	0.2
Smoothing noise clip	0.5
Length scale l	0.1

C.2 PGA-MAP-Elites

Table 3. PGA-MAP-Elites hyperparameters

Parameter	Value
Number of centroids	1024
Total batch size b	256
GA batch size b_{GA}	128
PG batch size b_{PG}	127
AI batch size b_{AI}	1
Policy networks	$[128, 128, \lvert\mathcal{A}\rvert]$
GA variation param. 1 σ_1	0.005
GA variation param. 2 σ_2	0.05
Actor network	$[128, 128, \lvert\mathcal{A}\rvert]$
Critic network	[256, 256, 1]
TD3 batch size N	100
Critic training steps n	3000
PG training steps m	150
Policy learning rate	5×10^{-3}
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Replay buffer size	10^6
Discount factor γ	0.99
Actor delay Δ	2
Target update rate	0.005
Smoothing noise var. σ	0.2
Smoothing noise clip	0.5

C.3 QD-PG

Table 4. QD-PG hyperparameters

Parameter	Value
Number of centroids	1024
Total batch size b	256
GA batch size b_{GA}	86
QPG batch size b_{PG}	85
DPG batch size b_{PG}	85
Policy networks	$[128, 128, \lvert\mathcal{A}\rvert]$
GA variation param. 1 σ_1	0.005
GA variation param. 2 σ_2	0.05
Actor network	$[128, 128, \lvert\mathcal{A}\rvert]$
Critic network	[256, 256, 1]
TD3 batch size N	100
Quality critic training steps n	3000
Diversity critic training steps n	300
PG training steps m	150
Policy learning rate	5×10^{-3}
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Replay buffer size	10^6
Discount factor γ	0.99
Actor delay Δ	2
Target update rate	0.005
Smoothing noise var. σ	0.2
Smoothing noise clip	0.5
Number nearest neighbors	5
Novelty scaling ratio	1.0

C.4 MAP-Elites

Table 5. MAP-Elites hyperparameters

Parameter	Value
Number of centroids	1024
Total batch size b	256
GA batch size b_{GA}	256
Policy networks	$[128, 128, \lvert\mathcal{A}\rvert]$
GA variation param. 1 σ_1	0.005
GA variation param. 2 σ_2	0.05

C.5 MAP-Elites-ES

Table 6. MAP-Elites-ES hyperparameters

Parameter	Value
Number of centroids	1024
Total batch size b	256
GA batch size b_{GA}	128
PG batch size b_{PG}	127
AI batch size b_{AI}	1
Policy networks	$[128, 128, \lvert\mathcal{A}\rvert]$
GA variation param. 1 σ_1	0.005
GA variation param. 2 σ_2	0.05
Number of samples	1000
Sample sigma	0.02

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009