Policy Gradient Assisted MAP-Elites

Olle Nilsson, Antoine Cully

Policy Gradient Assisted MAP-Elites

Olle Nilsson
Imperial College London
London, United Kingdom
olle.nilsson19@imperial.ac.uk

Antoine Cully
Imperial College London
London, United Kingdom
a.cully@imperial.ac.uk

ABSTRACT

Quality-Diversity optimization algorithms such as MAP-Elites, aim to generate collections of both diverse and high-performing solutions to an optimization problem. MAP-Elites has shown promising results in a variety of applications. In particular in evolutionary robotics tasks targeting the generation of behavioral repertoires that highlight the versatility of robots. However, for most robotics applications MAP-Elites is limited to using simple open-loop or low-dimensional controllers. Here we present Policy Gradient Assisted MAP-Elites (PGA-MAP-Elites), a novel algorithm that enables MAP-Elites to efficiently evolve large neural network controllers by introducing a gradient-based variation operator inspired by Deep Reinforcement Learning. This operator leverages gradient estimates obtained from a critic neural network to rapidly find higher-performing solutions and is paired with a traditional genetic variation to maintain a divergent search behavior. The synergy of these operators makes PGA-MAP-Elites an efficient yet powerful algorithm for finding diverse and high-performing behaviors. We evaluate our method on four different tasks for building behavioral repertoires that use deep neural network controllers. The results show that PGA-MAP-Elites significantly improves the quality of the generated repertoires compared to existing methods.

CCS CONCEPTS

• **Computing methodologies** → **Evolutionary robotics**.

KEYWORDS

Quality-Diversity, MAP-Elites, Neuroevolution

ACM Reference Format:
Olle Nilsson and Antoine Cully. 2021. Policy Gradient Assisted MAP-Elites. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3449639.3459304

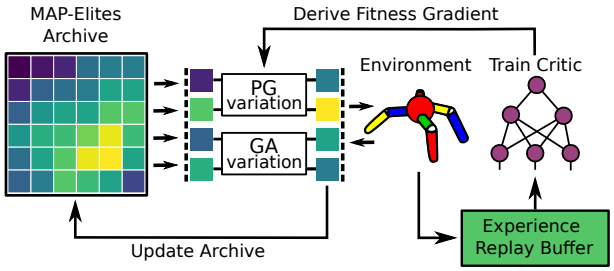

Figure 1: The conceptual approach behind PGA-MAP-Elites. While executing a standard MAP-Elites loop of repeated selection, variation and evaluation of solutions, the variation is split between two independent operators: 1) A Policy Gradient (PG) variation for directed performance improvement. 2) A genetic (GA) variation for divergent search. By training a critic neural network asynchronously to the MAP-Elites loop using experience collected during evaluations, performance gradients can swiftly be derived for any solution.

1 INTRODUCTION

Diversity is a catalyst of life. By finding a novel adaptation to the environment, species can thrive while being neither the fastest, strongest nor tallest globally [31]. This notion inspired researchers in Evolutionary Computation (EC) to pursue Quality-Diversity (QD) optimization [15, 43, 44]. In QD optimization, performance-based competition is considered only locally between solutions characterized as similar. Rather than optimizing strictly for a single best-performing solution, QD optimization algorithms aim to return a collection of solutions that are both as diverse as possible and as high-performing as possible. In robotics, this allows learning a repertoire of behaviors which is useful since this provides alternatives if one behavior suddenly becomes ineffective due to changes in the environment or damage to the robot [13, 47, 48]. In many cases, it is also desirable for a robot to discover the entire range of behaviors it is capable of rather than just a single behavior that maximizes a certain objective [11, 16]. Greedily optimizing a given objective may also cause the optimization process to prematurely converge to a local optimum, while simultaneously searching for diverse behaviors can help to find stepping stones that overcome local optima and lead to finding globally more optimal behaviors [17, 32].

QD optimization algorithms such as Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [13, 40, 48], are traditionally driven by a Genetic Algorithm (GA) for their capability of diversifying the search. This reliance on GAs limits the applicability of MAP-Elites to problems of low dimensionality. Typically the number of optimized parameters is kept below 100 [13, 15, 40]. GAs are also inefficient [12, 20] and prone to finding unstable solutions located on narrow peaks in the optimization landscape that are not repeatable in stochastic environments [14, 19, 25].

Deep Reinforcement Learning (DRL) [37–39] algorithms are based on an opposing methodology where a single performance-maximizing behavior is sought. In DRL, behaviors are learned via a

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459304

deep neural network (DNN) controller that is trained to predict the "optimal" action—the action that will most likely lead to maximizing the defined objective—to take given an observation. DRL leverages the function approximation strength of DNNs and powerful gradient-based training techniques, such as backpropagation [35], to guide the learning process directly towards improving performance. Using these techniques, DRL algorithms can solve problems in robotics that require the complex and precise control only achievable by DNNs with tens of thousands of parameters, in stochastic environments where learning robust behaviors is essential [22, 33].

This paper introduces the Policy Gradient Assisted MAP-Elites (PGA-MAP-Elites) algorithm, an extension of MAP-Elites which incorporates gradient-based optimization via a method based on DRL algorithm Twin Delayed Deep Deterministic policy gradient (TD3) [22]. By evaluating PGA-MAP-Elites on a set of stochastic behavior generation tasks requiring robots to be controlled by large DNN, we show that PGA-MAP-Elites successfully scales the generation of behavioral repertoires to new domains where current versions of MAP-Elites fail. In these tasks PGA-MAP-Elites achieves a powerful illumination of the search space, finding high-performing and robust solutions across the entire range of possible behaviors, where the highest performing solutions found rival those of modern DRL algorithms. The benchmark tasks used to evaluate PGA-MAP-Elites have been made available as the OpenAI Gym [5] based open-source module QDgym (https://github.com/ollenilsson19/QDgym). This module does not rely on any proprietary software, making our benchmarks easy to use for other researchers.

2 BACKGROUND AND RELATED WORK

2.1 Quality-Diversity Optimization

In contrast to standard optimization algorithms, which search for a single global solution regardless of its characteristics, QD optimization algorithms differentiate between solutions that have different characteristics. The domain that characterizes solutions is called the behavioral space [15] and is defined based on some property of solutions that is meaningful for the type of diversity sought. Characterization of a solution is called a behavioral descriptor (BD), denoted $\mathbf{b}$, that measures where in the behavioral space a solution lies. For example, if the task is for a robot to discover the range of gaits it is capable of, the BD may be the proportion of time each foot of the robot is in contact with the ground in a gait [7, 13, 48]. The task of QD optimization algorithms is thus to produce a collection of solutions that are as diverse as possible based on the defined behavioral space, and as high-performing as possible in its local region of the behavioral space. Each solution only competes based on performance with solutions that have similar BDs, or in other words, exist within the same niche of behavior.

2.2 MAP-Elites Algorithm

MAP-Elites is a simple but effective QD optimization algorithm that has been used to teach robots how to adapt to damage [13, 47, 48], generate aerodynamic designs [23] and to create content for games [1, 2]. In MAP-Elites, the behavioral space is discretized into a grid, which forms an archive for storing solutions where each cell corresponds to a behavioral niche. The goal of the algorithm is to return an archive with a maximum number of cells filled, where the solution in each cell is the highest performing solution possible within that niche. The MAP-Elites algorithm follows a repeated loop of uniformly selecting solutions from the current archive, applying a variation to the selected solutions to form new ones and evaluating the new solutions for addition to the archive. Solutions are added to the archive based on the status of the cell a solution's BD falls within. If that cell is empty the solution is added. If that cell is not empty, the solution replaces the one currently in that cell if the new solution has a higher performance based on a defined fitness function. For robotics applications, solutions are vectors $\mathbf{x}$ that parameterize controllers (such as neural networks) to control a robot with the goal of solving a certain task. The selection and variation are repeated until I solutions have been evaluated and the final archive is returned as the solution of the algorithm. To initialize the MAP-Elites algorithm an empty archive is first generated. The archive is represented by $\mathcal{X}$, that stores the parameter vector of the solution in each cell c, and $\mathcal{P}$, that stores the corresponding fitness value p. The archive is then initialized by G random solutions before entering the selection and variation loop. The MAP-Elites algorithm is further detailed in Alg. 1.

Algorithm 1 MAP-Elites algorithm. Adapted from [48].

procedure MAP-Elites( )
  $(\mathcal{X}, \mathcal{P}) \longleftarrow$ create_empty_archive()
  $i = 0$
  while $i < I$ **do** ▷ *Main Loop: I evaluations, batch-size b*
    if $i < G$ **then** ▷ *Initialization: G random* $\mathbf{x}$.
      $\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_b$ = random_solutions(b)
    else ▷ *Selection and variation*
      $\mathbf{x}_1, \ldots, \mathbf{x}_b$ = selection($\mathcal{X}, b$) ▷ *Uniform sampling*
      $\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_b$ = variation($\mathbf{x}_1, \ldots, \mathbf{x}_b$)
    i += b
    ADD_TO_ARCHIVE ($\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_b, \mathcal{X}, \mathcal{P}$)
  return archive $(\mathcal{X}, \mathcal{P})$
procedure ADD_TO_ARCHIVE(*Solution-List*, $\mathcal{X}, \mathcal{P}$)
  for $\mathbf{x}$ **in** *Solution-List* **do**
    $(p, \mathbf{b}) \longleftarrow$ evaluate($\mathbf{x}$)
    $c \longleftarrow$ get_cell_index($\mathbf{b}$)
    if $\mathcal{P}(c) = empty$ **or** $\mathcal{P}(c) < p$ **then**
      $\mathcal{P}(c) \longleftarrow p, \mathcal{X}(c) \longleftarrow \mathbf{x}$

By using a GA as the variation operator, MAP-Elites achieves a divergent search indirectly via the behavior-based archiving and a variation agnostic to the fitness objective. This divergent search methodology may cause inefficiencies or failure to learn. Even in problems where only a few parameters are optimized, the GA variation causes a slow convergence [20]. In problems that require behaviors to be encoded by DNNs with a large number of parameters, MAP-Elites with a GA variation typically fails to find the locally optimal behaviors due to a lack of directed search power [7]. In stochastic tasks the archiving methodology can cause inefficiencies if the behaviors found are not robust, as the expected fitness and BD of each solution need to be calculated by averaging over several evaluations. This averaging requirement can reduce

data-efficiency by several orders of magnitude [14, 19, 25]. In sequential decision-making tasks, where each evaluation may require thousands of costly simulation steps, MAP-Elites can thus become resource-demanding and challenging to employ.

An extension of MAP-Elites called Centroidal Voronoi Tessellation MAP-Elites (CVT-MAP-Elites) [48] automates the archive creation by spreading k cell-centroid locations maximally in behavioral space and is used as the basis for developing PGA-MAP-Elites.

2.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) for robotics applications is formalized as a Markov Decision Process (MDP) that considers a robot acting sequentially in an environment at discrete time-steps t. A deep neural network (DNN) controller with parameters ϕ encodes a behavior/policy $\pi_\phi(s)$, that for each observed state s_t, chooses an action a_t, leading to a new state s_{t+1}. Each such transition is scored by a reward function $r(s_t, a_t)$. The constraints of an MDP requires that the observation made at each time-step fully determines the robot's current state. The objective is to learn, from experience, the policy that maximizes the expected return $J(\phi) = \mathbb{E}[R_0]$ over a lifetime of T time-steps. The return is defined as $R_t = \sum_{i=t}^{T} \gamma^{i-t}\, r(s_i, a_i)$, where $\gamma \in [0, 1]$ is the discount factor which regulates importance of future rewards.

2.4 TD3 Algorithm

The TD3 algorithm [22] is one of the state-of-the-art methods in DRL for robotics. TD3 uses an actor-critic methodology to learn a policy (actor) indirectly via maximization of the action-value function $Q^\pi(s, a) = \mathbb{E}[R_t \mid s, a]$, approximated by a pair of DNNs (critics) with parameters θ_1 and θ_2. The action-value function encodes the expected return from being in state s and taking action a, thereafter following the policy π. The policy that maximizes the action-value function at every time-step thus maximizes the return. While acting in the environment, TD3 stores experience in the form of transitions $(s_t, a_t, r(s_t, a_t), s_{t+1})$ in a replay buffer $\mathcal{B}$ [34]. This collection of experience is used to derive a performance gradient for the policy. TD3 uses deterministic policies and calculates their performance gradient via the deterministic policy gradient [46],

$$\nabla_\phi J(\phi) = \mathbb{E}\left[\nabla_\phi \pi_\phi(s)\ \nabla_a Q_{\theta_1}(s, a)\big|_{a=\pi_\phi(s)}\right]. \tag{1}$$

To calculate this gradient, the critics are trained to approximate the action-value function via the Bellman equation [4], which describes the relationship between the action-value of one state-action pair and the action-value of the subsequent state-action pair,

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}\left[Q^\pi(s_{t+1}, \pi(s_{t+1}))\right]. \tag{2}$$

The action-value function can thus be learned by bootstrapping from the critics' current estimates to learn a better approximation. As the policy is deterministic, the expectation in Eq. 2 depends only on the environment and the action-value function can be learned off-policy [51] from experience collected by acting in the environment under any policy and stored in a replay buffer. TD3 calculates the update target in this bootstrapping by taking the minimum action-value prediction between the two critics to avoid overestimation feeding further overestimation and leading to instabilities. To further promote stability, dedicated target networks

Algorithm 2 TD3 algorithm. Adapted from [22].

$\pi_\phi \longleftarrow$ initialize_actor_network()
$Q_{\theta_1}, Q_{\theta_2} \longleftarrow$ initialize_critic_networks()
$\pi_{\phi'} \leftarrow \pi_\phi$, $Q_{\theta_1'}, Q_{\theta_2'} \leftarrow Q_{\theta_1}, Q_{\theta_2}$ ▷ *Initialize target networks*
$\mathcal{B} \longleftarrow$ initialize_replay_buffer()
$s_t \sim p(s_1)$ ▷ *Sample initial state*
for $t = 1 \rightarrow n\_iter$ **do** ▷ *Training Loop: n_iter iterations*
  $a_t \longleftarrow \pi_\phi(s_t) + \mathcal{N}(0, \sigma_a)$ ▷ *Select action and add noise*
  Apply action and observe $r(s_t, a_t)$ *and* s_{t+1}
  add_to_replay_buffer $((s_t, a_t, r(s_t, a_t), s_{t+1}), \mathcal{B})$
  Sample N transitions $(s_t, a_t, r(s_t, a_t), s_{t+1})$ *from* $\mathcal{B}$
  $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_p), -c, c)$ ▷ *Sample policy smoothing noise*
  $a_{t+1} \longleftarrow \pi_{\phi'}(s_{t+1}) + \epsilon$ ▷ *Predict next action and add noise*
  $y = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta_i'}(s_{t+1}, a_{t+1})$ ▷ *Calculate targets*
  $\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum \left(y - Q_{\theta_i}(s, a)\right)^2$ ▷ *Update critics*
  if t mod d **then**
    Update actor 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 critics targets*
    $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$ ▷ *Update actor target*

$(Q_{\theta_i'}, \pi_{\phi'})$ [38, 39] are used for both the critics and the actor to keep the update target stable between iterations. By minimizing the loss, $L(\theta_1, \theta_2) = \left(y - Q_{\theta_1}(s_t, a_t)\right)^2 + \left(y - Q_{\theta_2}(s_t, a_t)\right)^2$ over a batch of N transitions of experience, TD3 updates the critics by gradient descent towards the target y,

$$y = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta_i'}\left(s_{t+1}, \pi_{\phi'}(s_{t+1}) + \epsilon\right). \tag{3}$$

The next action is predicted as $a_{t+1} = \pi_{\phi'}(s_{t+1}) + \epsilon$, where $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_p), -c, c)$ is sampled Gaussian noise with variance σ_p clipped to a maximum magnitude c. The additional noise enforces higher action-values to actions more resistant to perturbations, promoting robustness in stochastic environments. Every d iterations the policy is updated via an approximation of Eq. 1 over a batch of N transitions of experience and the target networks updated by a factor τ to slowly track the main networks,

$$\begin{aligned} \theta_i' &\leftarrow \tau\theta_i + (1-\tau)\theta_i', \\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi'. \end{aligned} \tag{4}$$

As the polices are deterministic, exploration is achieved by adding Gaussian noise with a variance σ_a to the selected action as the robot interacts with the environment, $a_t = \pi_\phi(s_t) + \mathcal{N}(0, \sigma_a)$. The TD3 algorithm is further detailed in Alg. 2.

2.5 Related Work

Several recent algorithms explore creating fruitful synergies between DRL and EC methods for single objective search [26–28, 36, 50]. Most notably, CEM-RL [42] shows that pairing TD3 with population-based methods can improve performance in DRL benchmarks for robotics. In QD-based search, MAP-Elites with Evolution Strategies (MAP-Elites-ES) [7] is a recent algorithm that attempts to scale up MAP-Elites for use with high-dimensional controllers

represented by DNNs. By pairing MAP-Elites with the Evolution Strategies (ES) optimization method proposed in [45], MAP-Elites-ES create variation operators for directly optimizing a fitness objective and for directly optimizing a diversity objective based on the behaviors found so far. These operators optimize their respective objectives by finding empirical gradient estimates from sampling and evaluating a range of perturbations around a current solution. Using this method MAP-Elites-ES is able to find far higher-performing behaviors than MAP-Elites with GA variation when building repertoires for robots controlled by large DNNs. However, calculating gradients by sampling can require a large number of evaluations when optimizing DNNs. MAP-Elites-ES therefore typically finds repertoires containing far fewer behaviors than MAP-Elites with GA variation given an equal amount of experience in the environment [7]. This means MAP-Elites-ES can become computationally resource-demanding and impractical to use for building behavior repertoires with large DNN controllers.

Recently, works in DRL have incorporated select ideas from QD optimization such as using archives to collect solutions [30], using GAs for exploration with neural networks [41] or solving several tasks from a single learning process [24]. Although most common is the incorporation of diversity to improve exploration [3, 6, 8, 9, 17, 18, 21]. In this last category, the recent QD-RL [6] algorithm is of particular interest. QD-RL searches for diversity directly as a way to overcome deceptive rewards in DRL problems by introducing a policy gradient for diversity. This method requires defining an additional behavior characterization at the time-step level, called a "state-BD", that must be a valid state description in an MDP and simultaneously provide a meaningful space for searching for the type of diversity sought. QD-RL pairs this search for diversity with a standard PG method optimizing for performance, and maintains a population from which solutions are sampled for optimization based on the Pareto-front of the two measures. Replacing this population with a MAP-Elites archive would allow evaluating the QD-RL approach for the generation of behavior repertoires and could reveal a potent QD optimization algorithm.

3 PGA-MAP-ELITES

Policy Gradient Assisted MAP-Elites (PGA-MAP-Elites) is an extension of MAP-Elites that targets evolving DNN controllers by combining the search power and data-efficiency of Policy Gradient methods with the exploration capabilities of Genetic Algorithms. The main concept of PGA-MAP-Elites is illustrated in Fig. 1. While following the usual MAP-Elites loop, PGA-MAP-Elites uses two independent variation operators in parallel: 1) a Policy Gradient (PG) operator. 2) a standard Genetic Algorithm (GA) operator. At each iteration of the MAP-Elites loop, PGA-MAP-Elites collects the experience from evaluating controllers to train a pair of critic neural networks based on the methods of the TD3 algorithm. These trained critics are used to derive fitness gradient approximations for the PG variation, which updates controllers towards selecting actions that maximize the first critic's action-value predictions.

3.1 Training the Critics

More formally, PGA-MAP-Elites derives fitness gradients by training two critic neural networks, Q_{θ_1} and Q_{θ_2}, to approximate the action-value function. A single "greedy" controller is trained together with the critics to predict the action with the maximum action-value for calculation of the critic update target in the same way the actor is used in TD3 (see Eq. 3). However, unlike the actor in TD3 the greedy controller does not directly interact with the environment. The greedy controller is denoted π_{ϕ_c} and is a neural network with parameters ϕ_c, of the same architecture as the controllers evolved in the MAP-Elites loop. For each iteration of PGA-MAP-Elites, controllers are sampled from the archive, modified according to the variation operator and re-evaluated for addition to the archive. New experience from controller evaluations is collected in a replay buffer $\mathcal{B}$. The replay buffer has a limited maximum size and old experience is overwritten on a first-in-first-out basis. Asynchronous to each iteration of selection, variation and evaluation, the critics are trained for *n_crit* steps of gradient descent. For each training step, the parameters of both critics are updated such that the average action-value prediction of both critics tend towards the target y. Each training step thus aims to minimize the loss $L(\theta_1, \theta_2) = \left(y - Q_{\theta_1}(s_t, a_t)\right)^2 + \left(y - Q_{\theta_2}(s_t, a_t)\right)^2$, averaged over N transitions of experience sampled uniformly from the replay buffer. The target is calculated in the same way as in TD3, via the Bellman equation (Eq. 2) and taking the minimum action-value prediction between the two critics,

$$y = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta_i'}\left(s_{t+1}, \pi_{\phi_c'}(s_{t+1}) + \epsilon\right). \quad (5)$$

The subsequent action is predicted by the greedy controller as $\pi_{\phi_c'}(s_{t+1})$, with the addition of sampled noise ϵ, carried over from

Algorithm 3 PGA-MAP-Elites algorithm. Uses notation from [48].

procedure PGA-MAP-Elites
  $(\mathcal{X}, \mathcal{P}) \longleftarrow$ create_empty_archive()
  $Q_{\theta_1}, Q_{\theta_2}, \pi_{\phi_c} \longleftarrow$ initialize_critic_networks()
  $Q_{\theta_1'}, Q_{\theta_2'} \leftarrow Q_{\theta_1}, Q_{\theta_2}$ ▷ *Initialize target networks*
  $\pi_{\phi_c'} \leftarrow \pi_{\phi_c}$ ▷ *Initialize target network*
  $\mathcal{B} \longleftarrow$ initialize_replay_buffer()
  $i = 0$
  while $i < I$ **do** ▷ *Main Loop: I evaluations, batch-size b*
    if $i < G$ **then** ▷ *Initialization: G random π_ϕ.*
      $\pi_{\widehat{\phi}_1}, \ldots, \pi_{\widehat{\phi}_b} =$ random_solutions(b)
    else ▷ *Selection and variation*
      $\pi_{\widehat{\phi}_1} =$ TRAIN_CRITIC$(Q_{\theta_i}, Q_{\theta_i'}, \pi_{\phi_c}, \pi_{\phi_c'}, \mathcal{B})$ i=1,2
      $\pi_{\widehat{\phi}_2}, \ldots, \pi_{\widehat{\phi}_b} =$ VARIATION$(b-1, \mathcal{X}, Q_{\theta_1}, \mathcal{B})$
    ADD_TO_ARCHIVE $\left(\pi_{\widehat{\phi}_1}, \ldots, \pi_{\widehat{\phi}_b}, \mathcal{X}, \mathcal{P}, \mathcal{B}\right)$
    i += b
  return archive $(\mathcal{X}, \mathcal{P})$
procedure ADD_TO_ARCHIVE(*Controller-List*, $\mathcal{X}, \mathcal{P}, \mathcal{B}$)
  for π_ϕ **in** *Controller-List* **do**
    $(p, \mathbf{b}, \text{transitions}) \longleftarrow$ evaluate(π_ϕ)
    add_to_replay_buffer (transitions, $\mathcal{B}$)
    $c \longleftarrow$ get_cell_index($\mathbf{b}$)
    if $\mathcal{P}(c) = empty$ **or** $\mathcal{P}(c) < p$ **then**
      $\mathcal{P}(c) \longleftarrow p, \mathcal{X}(c) \longleftarrow \pi_\phi$

Algorithm 4 PGA-MAP-Elites Critic Training

procedure TRAIN_CRITIC($Q_{\theta_1}, Q_{\theta_2}, Q_{\theta'_1}, Q_{\theta'_2}, \pi_{\phi_c}, \pi_{\phi'_c}, \mathcal{B}$)
  for $t = 1 \rightarrow n\_crit$ **do** ▷ *Training Loop: n_crit iterations*
    Sample N transitions $(s_t, a_t, r(s_t, a_t), s_{t+1})$ *from* $\mathcal{B}$
    $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_p), -c, c)$ ▷ *Sample smoothing noise*
    $y = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta'_i}\left(s_{t+1}, \pi_{\phi'_c}(s_{t+1}) + \epsilon\right)$
    $\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum \left(y - Q_{\theta_i}(s,a)\right)^2$ ▷ *Update Critics*
    if $t \bmod d$ **then**
      Update greedy controller using gradient descent
      $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_\phi \pi_{\phi_c}(s_t)\ \nabla_a Q_{\theta_1}(s_t, a)\big|_{a=\pi_{\phi_c}(s_t)}$
      $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$ ▷ *Update targets*
      $\phi'_c \leftarrow \tau\phi_c + (1-\tau)\phi'_c$ ▷ *Update target*
  return controller π_{ϕ_c}

TD3. This noise addition is critical as it implicitly favors robust behaviors. The PG variation operator will thus be less prone to converging to behaviors that exist on narrow peaks in the fitness landscape which increases robustness in stochastic environments. In Eq. 5, $Q_{\theta'_i}$ and $\pi_{\phi'_c}$ denotes separate target neural networks for the critics and greedy controller that are used in the target calculation to promote stability. The target networks are updated to slowly track the main networks following Eq. 4. Every d training steps the greedy controller is updated towards choosing the actions that maximize the action-value predictions of the critic Q_{θ_1}. This update is calculated via the deterministic policy gradient (Eq. 1) based on N transitions of experience sampled uniformly from the replay buffer. Both this update and the critic update are applied using the Adam optimizing method [29].

The critic training procedure is further detailed in Alg. 4, and is called in Alg. 3 Line 12. This call represents the synchronization of the critic training with the MAP-Elites loop to transfer information between the processes. Experience from the latest batch of controller evaluations is sent to the critic training and the current state of the critic Q_{θ_1} and replay buffer is sent to be used in the PG variation. The current state of the greedy controller is added for evaluation as this controller may provide a useful behavior. Delaying the start of the critic training until after initialization ensures a good initial distribution of experience in the replay buffer.

3.2 Variation Operator

After each iteration of the algorithm, the current state of the critic Q_{θ_1} and replay buffer $\mathcal{B}$ are used to derive the PG-based variation for the subsequent iteration. This variation applies *n_grad* consecutive steps of gradient descent to a controller based on maximizing the action-value predictions of the critic Q_{θ_1}. For each step, the deterministic policy gradient (Eq. 1) is calculated (over a batch of N transitions sampled uniformly from the replay buffer) and applied using the Adam optimizing method [29]. In practice, this gradient is calculated by maximizing $Q_{\theta_1}(s, \pi_\phi(s))$ over the batch of transitions by back-propagating gradients of this expression w.r.t the controller parameters ϕ. This means fitness gradient estimates are obtained effectively immediately for any controller, without requiring any environment interaction by the controller the fitness gradient is being calculated for.

Algorithm 5 PGA-MAP-Elites Variation Operator

procedure VARIATION($batch\_size, \mathcal{X}, Q_{\theta_1}, \mathcal{B}$)
  for $i = 1 \rightarrow batch\_size$ **do**
    if $i \leq n\_evo$ **then**
      $\pi_{\phi_a}, \pi_{\phi_b} = \text{selection}(\mathcal{X})$ ▷ *Uniform sampling*
      $\pi_{\widehat{\phi}_i} = \text{VARIATION\_GA}(\pi_{\phi_a}, \pi_{\phi_b})$
    else
      $\pi_\phi = \text{selection}(\mathcal{X})$ ▷ *Uniform sampling*
      $\pi_{\widehat{\phi}_i} = \text{VARIATION\_PG}(\pi_\phi, Q_{\theta_1}, \mathcal{B})$
  return controllers $\pi_{\widehat{\phi}_1}, \ldots, \pi_{\widehat{\phi}_{\text{batch\_size}}}$
procedure VARIATION_GA($\pi_{\phi_1}, \pi_{\phi_2}$)
  $\widehat{\phi} = \phi_1 + \sigma_1 \mathcal{N}(\mathbf{0}, \mathbf{I}) + \sigma_2 (\phi_2 - \phi_1)\, \mathcal{N}(0,1).$
  return controller $\pi_{\widehat{\phi}}$
procedure VARIATION_PG($\pi_\phi, Q_{\theta_1}, \mathcal{B}$)
  Update ϕ *to* $\widehat{\phi}$ *using gradient descent*
  for $i = 1 \rightarrow n\_grad$ **do** ▷ *n_grad steps of PG*
    Sample N transitions $(s_t, a_t, r(s_t, a_t), s_{t+1})$ *from* $\mathcal{B}$
    $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_\phi \pi_\phi(s_t)\ \nabla_a Q_{\theta_1}(s_t, a)\big|_{a=\pi_\phi(s_t)}$
  return controller $\pi_{\widehat{\phi}}$

To maintain the divergent search methodology of standard MAP-Elites the PG variation operator is paired with a GA variation operator. We specifically use the directional variation introduced in [49],

$$\widehat{\phi} = \phi_1 + \sigma_1 \mathcal{N}(\mathbf{0}, \mathbf{I}) + \sigma_2 (\phi_2 - \phi_1)\, \mathcal{N}(0,1). \tag{6}$$

The offspring controller parameters $\widehat{\phi}$ are created by adding Gaussian noise with a scalar covariance matrix $\sigma_1 \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the first parent controller's parameters ϕ_1, and displacing the parameter vector along the line from ϕ_1 towards the second parent controller's parameter vector ϕ_2. Displacement is decided by sampling a number from a Gaussian distribution with zero mean and variance σ_2.

The PG variation operator and the GA variation operator are combined in the function VARIATION, detailed in Alg. 5, and called in Alg. 3 Line 13. This function generates at each iteration the next batch of controllers to be evaluated. The number of controllers to sample is calculated as $b - 1$, where b is the total size of the batch, to make room for evaluation of the current state of the greedy controller. This VARIATION function returns a list of new controllers modified by either PG-based or GA-based variation, where the distribution between the two is determined by the parameter *n_evo*, typically set to create an equal proportion.

3.3 Implementation

The source code of PGA-MAP-Elites is available at (https://github.com/ollenilsson19/PGA-MAP-Elites) including containerized environments in which the experiments from the next section can be replicated. The implementation is based on source code from the authors of CVT-MAP-Elites [48] and authors of TD3 [22].

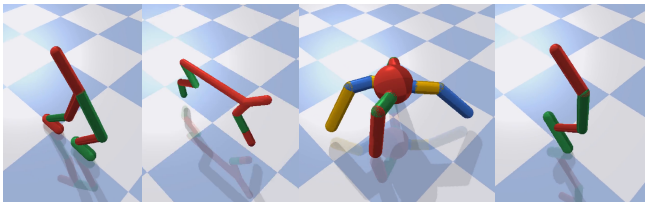

Figure 2: The evaluation tasks which extend the original PyBullet environments for generating behavioral repertoires.

4 EXPERIMENTAL EVALUATION

4.1 Evaluation Tasks

We evaluate PGA-MAP-Elites on four tasks for building behavior repertoires derived from standard PyBullet [10] DRL benchmarks for robotic locomotion. We call these tasks "QDWalker", "QDHalfCheetah", "QDAnt" and "QDHopper" (see Fig. 2). In each task, a simulated robot aims to discover all possible ways it can walk while maximizing a trade-off between speed and energy consumption. Fitness is defined as in the original PyBullet tasks by the accumulated forward progress made over the lifespan of the simulation (1000 steps) with an energy usage penalty and a reward for surviving each time-step of the simulation (see [10]). The progress made by a certain action in a certain state is attributed to that transition for training the critics. States are defined as the current center of gravity height, x, y and z velocity, roll, pitch and yaw angles and the relative position of the robot's joints. States thus have 22, 26, 28 and 15-dimensions for respective tasks. Actions are continuous-valued torques to apply to the robot's joints to control it. 6, 6, 8 and 3-dimensions for respective tasks. The BD used is the time proportion each foot of the robot is in contact with the ground in a behavior. This is a common definition also used in [7, 13, 48] for similar tasks. The BD thus has 2, 2, 4 and 1-dimension(s) for respective tasks. The tasks are stochastic in the sense that the initial joint-positions are sampled from a Gaussian distribution.

4.2 Comparisons

For comparison, we consider four algorithms: standard (CVT)-MAP-Elites [48] with directional variation (equivalent to setting $n\_evo = b$ in Alg. 5), MAP-Elites-ES [7], QD-RL [6] with a MAP-Elites archive and a version of TD3 [22] where a MAP-Elites archive is used to passively collect behaviors for comparison purposes. We also include a version of PGA-MAP-Elites using only PG variation by setting $n\_evo = 0$ in Alg. 5. We use implementations provided by the original authors of MAP-Elites-ES [7] and QD-RL[1] for the comparisons of these algorithms. Each task is repeated for 20 random seeds, over one-million controller evaluations.

4.3 Hyper-parameters

Hyper-parameters values used for PGA-MAP-Elites are given in Table 1. Common hyper-parameters are identical between standard MAP-Elites, TD3 and PGA-MAP-Elites and based on what worked best for MAP-Elites and TD3. We use neural network controllers with three layers. The first two layers have 128 neurons each and the number of neurons in the last layer equals the action dimension. This leads to the behavior being encoded by 20230 parameters in the QDWalker task, 20742 in QDHalfCheetah, 21256 in QDAnt and 18947 in QDHopper. We use this controller architecture to have the fewest parameters possible, but still enough to solve the tasks well. This was determined by running TD3 on the original PyBullet tasks with progressively smaller networks until a performance loss was observed. This choice gives standard MAP-Elites the best chance of finding high-performing behaviors. As these controllers have 3-4 times fewer parameters than used in the original MAP-Elites-ES paper [7], we reduce the sample-size in the gradient estimates from 10000 to 1000, allowing MAP-Elites-ES to make more additions to the archive to find the best repertoires possible within the computational resources available to us. Otherwise, hyper-parameters are those reported in the original papers for the damage adaptation task in MAP-Elites-ES and the Ant-Maze in QD-RL. The behavior space is discretized into 1024 niches (32 bins per dimension) in QDWalker and QDHalfCheetah, 1296 niches (6 bins per dimension) in QDAnt and 1000 niches in QDHopper.

[1]We use the source code of an improved work-in-progress version of QD-RL that trails a MAP-Elites archive for population management and was provided by its authors.

Table 1: Hyper-parameter values.

Parameter	Value
Neurons controller networks	[128, 128, action dim.]
Neurons critic networks	[256, 256, 1]
Nr. of evaluations (I)	10^6
Nr. of random init. (G)	500
Evaluation batch size (b)	100
Critic training steps (n_crit)	300
Critic training learning rates	3×10^{-4}
Networks training batch size (N)	256
Replay buffer max. size	10^6
Discount factor (γ)	0.99
Target networks update rate (τ)	0.005
Smoothing noise variance (σ_p)	0.2
Smoothing noise clip value (c)	0.5
Target networks update freq. (d)	2
Variation operators split (n_evo)	b * 0.5 = 50
PG variation steps (n_grad)	10
PG variation learning rate	0.001
GA variation param. 1 (σ_1)	0.005
GA variation param. 2 (σ_2)	0.05

4.4 Evaluation Metrics

We consider four main metrics for evaluation and report p-values based on the Wilcoxon rank-sum test.

- **QD-score** [43]: The total sum of fitness across all solutions in the archive. As fitness can be negative, the fitness of all behaviors is offset by the lowest fitness behavior found across all compared algorithms for a given task when calculating the QD-score. This avoids penalizing algorithms for discovering additional solutions.

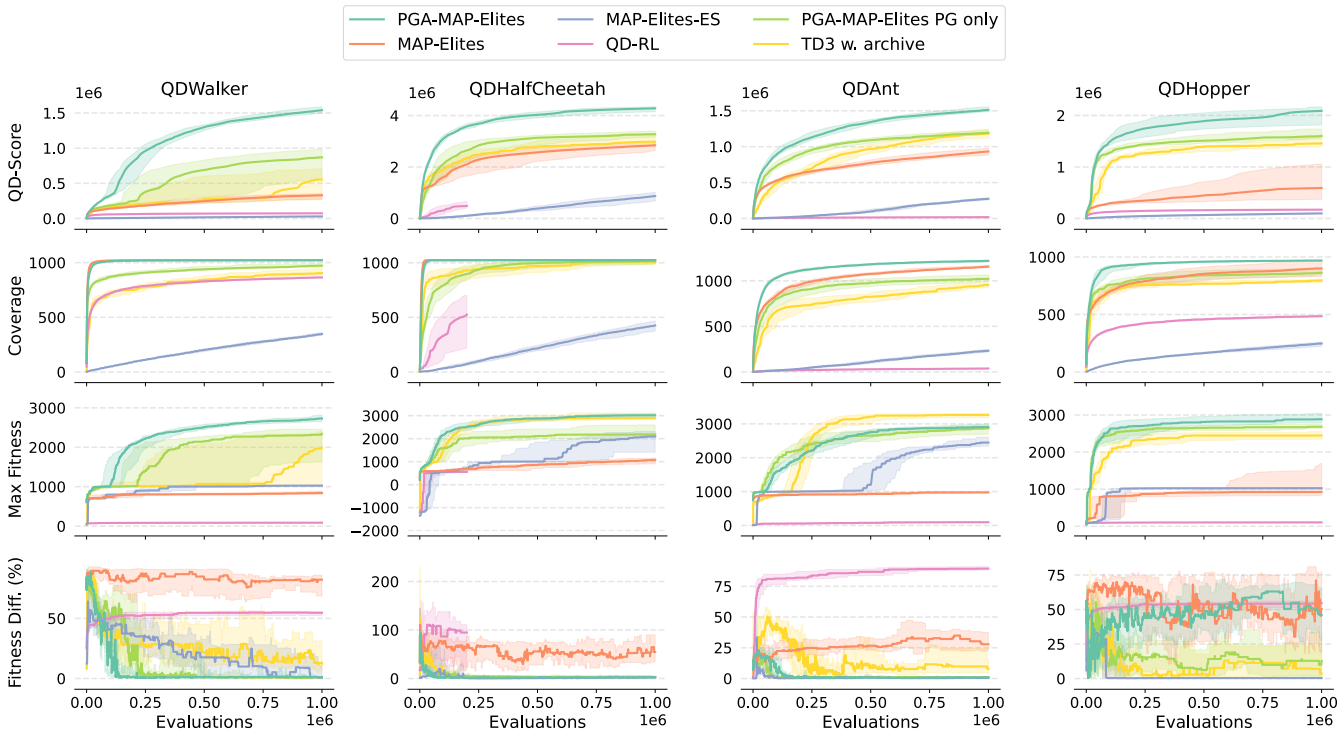

Figure 3: Results for the compared algorithms in each evaluation task. Each task is repeated for 20 runs with different random seeds, over one-million controller evaluations for each run. Each evaluation metrics is displayed with the median over the 20 seeds as a solid line with a shaded area around it bounded by the first and third quartiles.

- **Coverage**: The total number of solutions in the archive. Full coverage is achieved when a solution is found for each cell in the archive.
- **Max Fitness**: The overall fittest solution in the archive.
- **Fitness Difference**: To evaluate the robustness of each algorithm in stochastic environments, the fitness difference between a single evaluation and the fitness averaged over 10 evaluations for the current max fitness solution is considered. Each algorithm is only allowed a single evaluation for adding solutions to the archive which needs to be robust for the algorithm to be efficient in stochastic environments. We use the solution with max fitness for assessing robustness as it is likely the most sensitive to noise.

4.5 Experimental Results

Figure 3 shows the main evaluation metrics for each algorithm in each task. PGA-MAP-Elites achieves the best overall repertoire in all tasks as seen by the significantly higher QD-Scores compared to other algorithms ($p < 10^{-5}$). The coverage metric shows that PGA-MAP-Elites retains the divergent search capability of MAP-Elites and achieves similar or better ($p < 0.03$ for QDAnt and QDHopper) coverage in each task. Compared to other algorithms PGA-MAP-Elites has better coverage in all tasks ($p < 10^{-5}$). Although standard MAP-Elites achieves a good coverage in all tasks, the generated archives (Fig. 4) show that most behaviors found are of low fitness. PGA-MAP-Elites on the other hand is able to combine this good coverage with finding high-performing solutions across the behavior space (Fig. 4).

In the max fitness metric, PGA-MAP-Elites outperforms all other algorithms in the QDWalker and QDHopper tasks ($p < 0.01$). In QDHalfCheetah the max fitness difference is not statistically significant between PGA-MAP-Elites and TD3 but in QDAnt TD3 achieves a higher max fitness ($p < 10^{-6}$). It is not surprising that TD3 is able to find a few higher-performing behaviors in certain tasks as the critics in TD3 only have to estimate the action-value function well around the current policy for TD3 to continuously improve. In PGA-MAP-Elites, the critics need to estimate the action-value function over the entire behavior space which likely is more difficult. The archive plots (Fig. 4) confirm that TD3 can find best solution with a similar fitness to those found by PGA-MAP-Elites. However, PGA-MAP-Elites finds many more solutions in the medium to high fitness range leading to much better archives overall.

The fitness difference metric shows that PGA-MAP-Elites learns very robust behaviors in all tasks, except in QDHopper where the fitness difference (loss in this case) is large. The fitness difference for the PG only variant shows that the PG variation operator helps to improve sensitivity also in this task. This indicates that the sensitivity to noise exhibited by PGA-MAP-Elites in the QDHopper task is related to the GA variation operator. A solution to this could thus be to apply a step of PG variation to each solution after the GA variation has been applied.

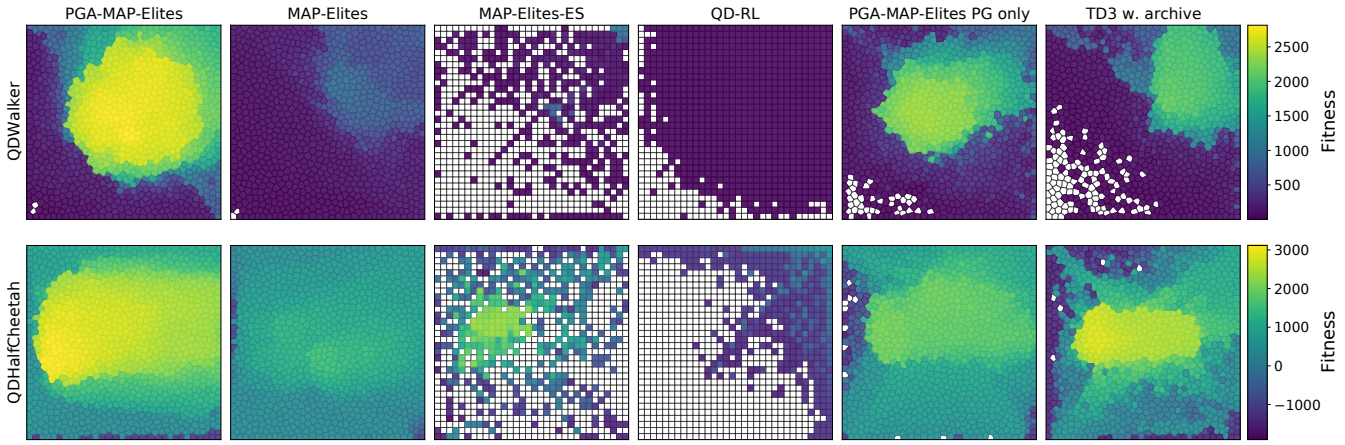

Figure 4: Typical archives found by each algorithm in the QDWalker and QDHalfCheetah tasks. In these tasks the archives form square grids as the BDs are 2-dimensional. Increasing feet contact time from left to right and from bottom to top.

MAP-Elites-ES finds good behaviors in the QDAnt and QDHalfCheetah tasks where it looks like it would eventually catch up with PGA-MAP-Elites given additional evaluations. However, despite finding a few good behaviors the overall repertoires found by MAP-Elites-ES have far fewer behaviors than other algorithms. MAP-Elites-ES therefore also lacks in the QD-Score metric. Figure 4 shows that the final archive for QDHalfCheetah is typically scattered but MAP-Elites-ES is able to find a number of high-performing behaviors in one local region of the behavior space. Given enough evaluations, MAP-Elites-ES would likely build strong repertoires in these two tasks. In the QDWalker and QDHopper tasks MAP-Elites-ES seemingly fails to find high-performing behaviors which may be related to the behavior space being relatively easy to cover and doing so does not correlate strongly with finding high-performing behaviors. The learning process of MAP-Elites-ES could thus be halted by its directed optimization for diversity, as finding behaviors that cover new regions of the behavior space is not necessarily meaningful for advancing the learning process.

QD-RL seemingly fails in all tasks which is likely related to the method used to directly optimize for diversity. Like in MAP-Elites-ES, the directed optimization for diversity may not be meaningful given that diversity in the BD and performance are not aligned. Moreover, as QD-RL optimizes for diversity based on an MDP, it assumes that the sum of the novelty of each state (based on the state-BD) provides a meaningful characterization for the novelty of the entire behavior. We define the state-BD at each time-step as the feet contact time up to that time-step as this is the space where we seek diversity. This is likely not appropriate for QD-RL. The ability to define a state-BD compatible with the commonly used BD considered in this paper remains an open question. QD-RL performs slightly better in QDHalfCheetah . In this task QD-RL finds behaviors able to survive the entire 1000 simulation steps without falling over, creating an increased resource demand for QD-RL in this task compared to other tasks. The use of only four agents in parallel (following the source code provided by the QD-RL authors) compared to 100 in PGA-MAP-Elites means our resources can not run QD-RL for more than 2×10^5 evaluations in QDHalfCheetah.

5 CONCLUSIONS

This paper present the PGA-MAP-Elites algorithm as an approach to building high-quality behavior repertoires for simulated robots where large DNN controllers are required. PGA-MAP-Elites introduces a PG-based variation to MAP-Elites for increased search power and data-efficiency and pairs it with the exploration capabilities of GA-based variation. The evaluation on four different repertoire-building tasks shows that PGA-MAP-Elites is able to efficiently build high-quality repertoires that significantly outperform those found by existing methods and successfully scales the use of MAP-Elites to evolving controllers with more than 20000 parameters.

PGA-MAP-Elites was developed to specifically target the generation of behavior repertoires for locomotion, but for future work we will investigate our approach for manipulation-based tasks and tasks where the BDs are aligned with performance to a high degree. Task with highly deceptive fitness structures are of particular interest as the MAP-Elites-ES and QD-RL approaches are likely to be more competitive in comparison in such tasks because of their ability to directly optimize for diversity.

This paper has shown clear benefits for using a PG variation with MAP-Elites. The benefits of using a PG variation come with the costs of the limitations of PG methods. In particular, the type of controllers that can be used with PGA-MAP-Elites are limited to differentiable function approximators such as neural networks. The setup of the PG variation is also limited to problems with full observability, satisfying the theoretical constraints of an MDP. These are two restrictions that do not apply to standard MAP-Elites variants.

ACKNOWLEDGMENTS

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/V006673/1 project REcoVER. We also want to thank the members of the Adaptive and Intelligent Robotics Lab at Imperial College London for their useful comments.

REFERENCES

[1] Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. 2020. Interactive Constrained MAP-Elites Analysis and Evaluation of the Expressiveness of the Feature Dimensions. arXiv:2003.03377 [cs.AI]

[2] Alberto Alvarez, Steve Dahlskog, José M. Font, and Julian Togelius. 2019. Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites. In *CoG*. IEEE, 1–8. http://dblp.uni-trier.de/db/conf/cig/cog2019.html#AlvarezDFT19

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, Vol. 2017-Decem. Neural information processing systems foundation, 5049–5059. arXiv:1707.01495

[4] Richard Bellman. 1954. The theory of dynamic programming. *Bull. Amer. Math. Soc.* 60, 6 (11 1954), 503–515. https://projecteuclid.org:443/euclid.bams/1183519147

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[6] Geoffrey Cideron, Thomas Pierrot, Nicolas Perrin, Karim Beguir, and Olivier Sigaud. 2020. QD-RL: Efficient Mixing of Quality and Diversity in Reinforcement Learning. arXiv:2006.08505 [cs.AI]

[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* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 67–75. https://doi.org/10.1145/3377930.3390217 Implementation: https://github.com/uber-research/Map-Elites-Evolutionary.

[8] Cedric Colas, Olivier Sigau, and Pierre Yves Oudeyer. 2018. GEP-PG: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *35th International Conference on Machine Learning, ICML 2018*, Vol. 3. International Machine Learning Society (IMLS), 1682–1691. arXiv:1802.05054

[9] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2018. Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 5032–5043.

[10] Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org. Implementation: https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/gym/pybullet_envs/gym_locomotion_envs.py.

[11] Antoine Cully. 2019. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, Inc, 81–89. https://doi.org/10.1145/3321707.3321804 arXiv:1905.11874

[12] Antoine Cully. 2020. Multi-Emitter MAP-Elites: Improving quality, diversity and convergence speed with heterogeneous sets of emitters. arXiv:2007.05352 [cs.NE]

[13] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* (2015), 503–507. https://doi.org/10.1038/nature14422 arXiv:1407.3501

[14] Antoine Cully and Yiannis Demiris. 2018. Hierarchical Behavioral Repertoires with Unsupervised Descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 69–76. https://doi.org/10.1145/3205455.3205571

[15] Antoine Cully and Yiannis Demiris. 2018. Quality and Diversity Optimization: A Unifying Modular Framework. *IEEE Transactions on Evolutionary Computation* (2018). https://doi.org/10.1109/TEVC.2017.2704781 arXiv:1708.09251

[16] Antoine Cully and Jean Baptiste Mouret. 2013. Behavioral repertoire learning in robotics. In *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*. https://doi.org/10.1145/2463372.2463399

[17] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2020. First return then explore. (apr 2020). arXiv:2004.12919 http://arxiv.org/abs/2004.12919

[18] Benjamin Eysenbach, Julian Ibarz, Abhishek Gupta, and Sergey Levine. 2019. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR. arXiv:1802.06070

[19] Manon Flageat and Antoine Cully. 2020. Fast and stable MAP-Elites in noisy domains using deep grids. *The 2020 Conference on Artificial Life* (2020). https://doi.org/10.1162/isal_a_00316

[20] Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. 2020. Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 94–102. https://doi.org/10.1145/3377930.3390232

[21] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. 2017. Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. (aug 2017). arXiv:1708.02190 http://arxiv.org/abs/1708.02190

[22] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 1587–1596. http://proceedings.mlr.press/v80/fujimoto18a.html Implementation: https://github.com/sfujim/TD3.

[23] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2017. Aerodynamic design exploration through surrogate-assisted illumination. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 3330.

[24] Tanmay Gangwani, Jian Peng, and Yuan Zhou. 2020. Harnessing Distribution Ratio Estimators for Learning Agents with Quality and Diversity. arXiv:2011.02614 [cs.LG]

[25] Jorge Gomes, Sancho Moura Oliveira, and Anders Lyhne Christensen. 2018. An approach to evolve and exploit repertoires of general robot behaviours. *Swarm and Evolutionary Computation* (2018).

[26] Abhishek Gupta, Russell Mendonca, Yu Xuan Liu, Pieter Abbeel, and Sergey Levine. 2018. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, Vol. 2018-Decem. Neural information processing systems foundation, 5302–5311. arXiv:1802.07245

[27] Rein Houthooft, Richard Y. Chen, Phillip Isola, Bradly C. Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. 2018. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, Vol. 2018-Decem. Neural information processing systems foundation, 5400–5409. arXiv:1802.04821

[28] Shauharda Khadka and Kagan Tumer. 2018. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, Vol. 2018-Decem. Neural information processing systems foundation, 1188–1200. arXiv:1805.07917

[29] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[30] Ayaka Kume, Eiichi Matsumoto, Kuniyuki Takahashi, Wilson Ko, and Jethro Tan. 2017. Map-based Multi-Policy Reinforcement Learning: Enhancing Adaptability of Robots by Deep Reinforcement Learning. (oct 2017). arXiv:1710.06117 http://arxiv.org/abs/1710.06117

[31] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation* 19, 2 (2011), 189–222. https://doi.org/10.1162/EVCO_a_00025

[32] Joel Lehman and Kenneth O. Stanley. 2011. Evolving a diversity of creatures through novelty search and local competition. In *Genetic and Evolutionary Computation Conference, GECCO'11*. 211–218. https://doi.org/10.1145/2001576.2001606

[33] 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 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR. arXiv:1509.02971

[34] Long-Ji Lin. 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Mach. Learn.* 8, 3–4 (May 1992), 293–321. https://doi.org/10.1007/BF00992699

[35] Seppo Linnainmaa. [n.d.]. Taylor Expansion of the Accumulated Rounding Error. 16, 2 ([n. d.]), 146–160. https://doi.org/10.1007/BF01931367

[36] Niru Maheswaranathan, Luke Metz, George Tucker, and Jascha Sohl-Dickstein. 2018. Guided evolutionary strategies: escaping the curse of dimensionality in random search. *arXiv* (2018), 1–16. https://doi.org/arXiv:1806.10230v2 arXiv:1806.10230

[37] Volodymyr Mnih, Adria Puigdomenech Badia, Lehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *33rd International Conference on Machine Learning, ICML 2016*, Vol. 4. International Machine Learning Society (IMLS), 2850–2869. arXiv:1602.01783

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (dec 2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[39] 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

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

[41] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. 2018. Parameter space noise for exploration. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR. arXiv:1706.01905

[42] Aloïs Pourchot and Olivier Sigaud. 2019. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR. arXiv:1810.01222

[43] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers Robotics AI* 3, JUL (jul 2016). https://doi.org/10.3389/frobt.2016.00040

[44] Justin K. Pugh, L. B. Soros, Paul A. Szerlip, and Kenneth O. Stanley. 2015. Confronting the challenge of quality diversity. In *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, Inc, 967–974. https://doi.org/10.1145/2739480.2754664

[45] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. (mar 2017). arXiv:1703.03864 http://arxiv.org/abs/1703.03864

[46] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *31st International Conference on Machine Learning, ICML 2014*.

[47] Danesh Tarapore, Jeff Clune, Antoine Cully, and Jean-Baptiste Mouret. 2016. How Do Different Encodings Influence the Performance of the MAP-Elites Algorithm?. In *Genetic and Evolutionary Computation Conference*.

[48] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2017. Using Centroidal Voronoi Tessellations to Scale Up the Multi-dimensional Archive of Phenotypic Elites Algorithm. *IEEE Transactions on Evolutionary Computation* (2017), 9. https://doi.org/10.1109/TEVC.2017.2735550 Implementation: https://github.com/resibots/pymap_elites.

[49] Vassilis Vassiliades and Jean-Baptiste Mouret. 2018. Discovering the Elite Hypervolume by Leveraging Interspecies Correlation. *Proceedings of the Genetic and Evolutionary Computation Conference* (2018).

[50] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. (nov 2016). arXiv:1611.05763 http://arxiv.org/abs/1611.05763

[51] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* (1992). https://doi.org/10.1007/bf00992698