PROXIMAL POLICY GRADIENT ARBORESCENCE FOR QUALITY DIVERSITY REINFORCEMENT LEARNING

Sumeet Batra
University of Southern California
Los Angeles, CA 90089
ssbatra@usc.edu

Bryon Tjanaka
University of Southern California
Los Angeles, CA 90089
tjanaka@usc.edu

Matthew C. Fontaine
University of Southern California
Los Angeles, CA 90089
mfontain@usc.edu

Aleksei Petrenko
University of Southern California
Los Angeles, CA 90089
petrenko@usc.edu

Stefanos Nikolaidis
University of Southern California
Los Angeles, CA 90089
nikolaid@usc.edu

Gaurav S. Sukhatme
University of Southern California
Los Angeles, CA 90089
gaurav@usc.edu

ABSTRACT

Training generally capable agents that thoroughly explore their environment and learn new and diverse skills is a long-term goal of robot learning. Quality Diversity Reinforcement Learning (QD-RL) is an emerging research area that blends the best aspects of both fields – Quality Diversity (QD) provides a principled form of exploration and produces collections of behaviorally diverse agents, while Reinforcement Learning (RL) provides a powerful performance improvement operator enabling generalization across tasks and dynamic environments. Existing QD-RL approaches have been constrained to sample efficient, deterministic *off-policy* RL algorithms and/or evolution strategies, and struggle with highly stochastic environments. In this work, we, for the first time, adapt on-policy RL, specifically Proximal Policy Optimization (PPO), to the Differentiable Quality Diversity (DQD) framework and propose additional improvements over prior work that enable efficient optimization and discovery of novel skills on challenging locomotion tasks. Our new algorithm, Proximal Policy Gradient Arborescence (PPGA), achieves state-of-the-art results, including a 4x improvement in best reward over baselines on the challenging humanoid domain.

1 INTRODUCTION

Quality Diversity (QD) algorithms enable the exploration and discovery of diverse skills in a behavior space. For example, a QD algorithm can train different locomotion gaits for a walker (Cully et al., 2015), discover different grasping trajectories for a manipulator (Morel et al., 2022), or generate a diverse range of human faces (Fontaine & Nikolaidis, 2021). However, since these algorithms are generally oriented towards solving exploration problems, they struggle to find performant policies in high-dimensional robot learning tasks. QD-RL is an emerging field that attempts to combine the principled exploration capabilities of QD with the powerful performance improvement capabilities of RL. Prior methods have leveraged *off-policy* RL, specifically TD3, to estimate the gradient of performance, and either Evolution Strategies (ES) or TD3 to estimate the gradient of diversity in order to search for diverse, high-quality policies. They have shown success in exploration problems and certain robot locomotion tasks (Nilsson & Cully, 2021; Pierrot et al., 2022; Tjanaka et al., 2022b). Nonetheless, there remains a gap in performance between QD-RL and standard RL algorithms on continuous control tasks. Furthermore, off-policy RL algorithms were not designed with massive parallelization in mind, and there is little literature that explores how to leverage modern massively-

parallelized simulators with these algorithms, whereas there are numerous works exploring on-policy RL in these regimes (Makoviychuk et al., 2021; Rudin et al., 2021; Handa et al., 2022; Batra et al., 2021; Huang et al., 2022).

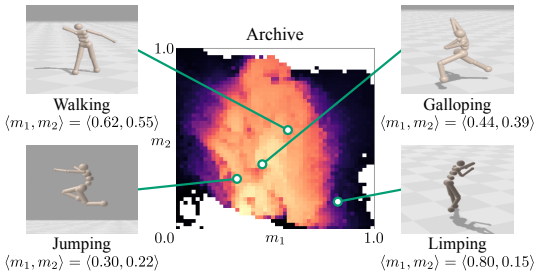


Figure 1: PPGA finds a diverse archive of high-performing locomotion behaviors for a humanoid agent by combining PPO gradient approximations with Differentiable Quality Diversity algorithms. The archive's dimensions correspond to the measures m_1 and m_2, i.e., the proportion of time that the left and right feet contact the ground. The color of each cell shows the objective value, i.e., how fast the humanoid moves. For instance, jumping moves the humanoid forward quickly, with the left and right feet individually contacting the ground 30% and 22% of the time, respectively.

From our investigation of prior methods, simply combining existing QD methods with an RL algorithm tends not to scale well to high-dimensional, highly dynamical systems such as Humanoid. For example, all QD-RL algorithms for locomotion to date use non-Markovian measures of behavioral diversity, which in many cases prevents direct RL-optimization. Most algorithms instead opt for policy parameter mutation, which struggles to scale well with deep neural networks. Prior methods that investigated combining Differentiable Quality Diversity and *off-policy* RL (Tjanaka et al., 2022b) achieved similar results as other baselines. However, given the gap in performance between standard RL and QD-RL algorithms in terms of best-performing policy, we believe that DQD algorithms, under a different formulation more synergistic with its underlying mechanisms, can close this gap. To this end, we leverage Proximal Policy Optimization (PPO) (Schulman et al., 2017), a popular *on-policy* RL algorithm, with Differentiable Quality Diversity (DQD) (Fontaine & Nikolaidis, 2021) because of the already present synergy. Specifically, DQD algorithms CMA-MEGA (Fontaine & Nikolaidis, 2021), and its more recent variation CMA-MAEGA (Fontaine & Nikolaidis, 2023), maintain a single search point (or policy in the case of RL) that moves through the behavior space and fills in new, unexplored regions with offspring policies constructed via gradient information *collected from online data*. It is through this high level view that we see the emergent synergy between PPO and DQD, in that PPO can be used to collect gradient estimates from online data when one or both of the objective and measure functions are Markovian and non-differentiable.

We make several key changes to CMA-MAEGA and PPO to maximally leverage their synergy. Our new algorithm, Proximal Policy Gradient Arborescence (PPGA), to the best of our knowledge, is the first QD-RL algorithm to not only achieve 4x performance in best reward on the humanoid domain, but achieve the *same* level of performance as PPO without sacrificing *any* of the diversity in the discovered policies. Specifically, we make the following contributions:

(1) We propose a vectorized implementation of PPO, VPPO, that jointly computes the objective and measure gradients with little overhead and without running separate PPO instances for each task **(2)** We generalize prior *CMA*-based DQD algorithms as instances of Natural Evolution Strategies (NES) and show that contemporary NES methods, specifically xNES, enable better training stability and performance for DQD algorithms **(3)** We introduce the notion of Markovian Measure Proxies (MMPs), which makes the typically non-Markovian measure functions used in QD-RL amenable to RL-optimization **(4)** We propose a new method to move the current search point, hereon referred to as the "search policy", to unexplored regions of the archive by iteratively "walking" it using collected online data and RL optimization of a novel multi-objective reward function.

2 BACKGROUND

2.1 DEEP REINFORCEMENT LEARNING

Reinforcement Learning algorithms search for a policy, a mapping of states to actions, that maximizes cumulative reward in an environment. RL assumes the discrete-time Markov Decision Process (MDP) formalism $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces respectively, $\mathcal{R}(s, a)$ is the reward function, $\mathcal{P}(s'|s, a)$ defines state transition probabilities, and γ is the discount factor. The RL objective is to maximize the discounted episodic return of a policy $\mathbb{E}\left[\sum_{k=0}^{T-1} \gamma^k R(s_k, a_k)\right]$ where T is episode length. **Deep Reinforcement Learning** solves the RL problem by finding a policy $\pi_\theta(a_t|s_t)$ parameterized by a deep neural network θ that represents a state-action mapping.

On-policy Deep RL methods directly learn the policy π_θ using experience collected by that policy or a recent version thereof. Contemporary methods (Mnih et al., 2016) fit the value function $V_\phi(s_t)$ to discounted returns and estimate the advantage $\hat{A}_t = \sum_{k=t}^{T-1} \gamma^k R(s_k, a_k) - V_\phi(s_t)$, which corresponds to the value of an action over the current policy (Schulman et al., 2016). From here, the gradient of the objective w.r.t. θ, or **policy gradient**, can be estimated as $\hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right]$, and the policy π_θ is trained using mini-batch gradient descent.

Trust region policy gradient Deep RL methods constrain the policy updates to maintain the proximity of π_θ to the *behavior* policy $\pi_{\theta_{old}}$ that was used to collect the experience. TRPO (Schulman et al., 2015) takes the largest policy improvement step that satisfies the strict KL-divergence constraint. Proximal Policy Optimization (PPO) (Schulman et al., 2017) approximates the trust region by optimizing a clipped surrogate objective where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the importance sampling ratio:

$$L(\theta) = \hat{\mathbb{E}}_{\pi_\theta}\left[\min(r_t(\theta)\hat{A}_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right].$$

Off-policy Deep RL algorithms learn parameterized state-action value functions $Q_\theta(s_t, a_t)$ that estimate the value of taking action a_t in state s_t. Then, actions are taken with a greedy policy $\arg\max_a Q_\theta(s_t, a_t)$, or an ε-greedy variation thereof. Q-functions can be learned from experience collected by recent or past versions of the policy or another policy altogether.

In continuous control problems, it can be difficult to find $a^* = \arg\max_a Q_\theta(s_t, a_t)$ due to an infinite number of possible actions. To work around this issue, off-policy methods such as DDPG (Lillicrap et al., 2016) and TD3 (Fujimoto et al., 2018) learn a deterministic policy $\mu_\phi(s_t)$ by solving $max_\phi(Q_\theta(s_t, \mu_\phi(s_t)))$ using gradient ascent. Other off-policy methods, such as soft actor-critic (SAC) Haarnoja et al. (2018), maintain an explicit policy π_θ, but similarly derive the policy gradient from the critic, allowing them to learn from off-policy data as well.

2.2 QUALITY DIVERSITY OPTIMIZATION

Unlike single-objective optimization methods such as RL, Quality Diversity algorithms search for an archive of high-performing, diverse policies. An optimal archive essentially answers the question, "how does performance change with behavior?" by mapping out the optimization landscape of a pre-defined behavior space. The QD problem (Chatzilygeroudis et al., 2021) assumes an *objective* function $f(\cdot)$ that quantifies the agent's performance and k *measure* functions $m_1(\cdot)...m_k(\cdot)$ that characterize the agent's behavior. The measure functions, represented jointly as $\mathbf{m}(\cdot)$, define an embedding the QD algorithm should span with diverse policies. The **QD objective** is to find a policy that maximizes f for every possible output of $\mathbf{m}$. However, the embedding formed by $\mathbf{m}$ is continuous, so the embedding space is discretized into a tessellation of M cells. The QD objective then becomes to maximize $\sum_{i=1}^{M} f(\theta_i)$, where θ_i is a policy whose measures $\mathbf{m}(\theta_i)$ fall in cell i of the tesselation.

QD algorithms originated with NSLC (Lehman & Stanley, 2011a;b) and MAP-Elites (Mouret & Clune, 2015; Cully et al., 2015). While these early QD algorithms built on genetic algorithms, modern QD algorithms incorporate optimization techniques like evolution strategies (Fontaine et al., 2020; Conti et al., 2018; Colas et al., 2020), gradient ascent (Fontaine & Nikolaidis, 2021; 2023), and differential evolution (Choi & Togelius, 2021). Several works have applied QD optimization to generative design (Hagg et al., 2020; Gaier et al., 2018), procedural content generation (Gravina

et al., 2019; Earle et al., 2022; Khalifa et al., 2018), robot manipulation (Morrison et al., 2020), and reinforcement learning (Nilsson & Cully, 2021; Tjanaka et al., 2022b; Pierrot & Flajolet, 2023).

2.3 Differentiable Quality Diversity

The Differentiable Quality Diversity (DQD) (Fontaine & Nikolaidis, 2021) algorithm Covariance Matrix Adaptation Map Elites via Gradient Arborescence (CMA-MEGA) considers the first-order QD problem where the objective and measure functions are differentiable, with gradients w.r.t. policy parameters represented as $\nabla f = \frac{\partial f}{\partial \theta}$ and $\nabla \mathbf{m} = \left[\frac{\partial m_1}{\partial \theta}, ..., \frac{\partial m_k}{\partial \theta}\right]$. CMA-MEGA maintains a *search policy* π_{θ_μ} in policy parameter space ($\theta_\mu \in \mathbb{R}^N$) corresponding to some cell in the archive given by the measures $< m_1(\pi_{\theta_\mu}), ..., m_k(\pi_{\theta_\mu}) >$, and a search distribution in objective-measure gradient coefficient space maintained by CMA-ES (Hansen, 2016), a zeroth-order optimizer that optimizes the coefficient distribution to produce coefficient vectors that point in the direction of *greatest archive improvement*. At a high level, CMA-MEGA branches off policies from the search policy in order to locally fill the archive, and then steps the search policy to new, unexplored regions of the archive. During the branching step, the gradients $< \nabla f, \nabla \mathbf{m} >_{\theta_\mu}$ and λ gradient coefficient vectors $< c_0, ..., c_k >_1, ..., < c_0, ..., c_k >_\lambda$ sampled from the CMA-ES search distribution $\mathbf{c}_i \sim \mathcal{N}(\mu, \Sigma) \in \mathbb{R}^{k+1}$ are combined via the dot product i.e. $< \nabla f, \nabla \mathbf{m} >_{\theta_\mu} \cdot \mathbf{c}_1, ...$ to produce local gradients $\nabla_1, ..., \nabla_\lambda$ around the search policy. Applying the gradients to the search policy gives us λ branched policies $\pi_{\theta_1}, ..., \pi_{\theta_\lambda}$. The new policies can then be ranked by how much they improve the archive, i.e., $f(\pi_{\theta_i}) - f(\pi_{\theta_{old}}), i \in [1, \lambda]$, where $\pi_{\theta_{old}}$ is the incumbent policy in the archive corresponding to the same cell as π_{θ_i}. Branched policies that map to new, unexplored cells in the archive have $f(\pi_{\theta_{old}})$ set to some minimum threshold. This implicitly biases the ranking towards the exploration of new, unvisited cells. This ranking is given to CMA-ES, which internally performs an update that steps the search distribution in the direction of the natural gradient w.r.t. greatest archive improvement. CMA-ES returns weights $w_1, ..., w_\lambda$ such that $\nabla_{step} =< w_1, ..., w_\lambda > \cdot < \nabla_1, ..., \nabla_\lambda >$ is the natural gradient in parameter space. This weighted linear recombination of the branching gradients is then used to step the search policy in the direction of greatest archive improvement $\theta_\mu \leftarrow \theta_\mu + \alpha \nabla_{step}$.

The current state-of-the-art DQD algorithm, Covariance Matrix Adaptation Map Annealing via Gradient Arborescence (CMA-MAEGA) (Fontaine & Nikolaidis, 2023), introduced the concept of soft archives to CMA-MEGA. Instead of maintaining the best policy in each cell, the archive maintains a threshold t_e and updates the threshold by $t_e \leftarrow (1 - \alpha)t_e + \alpha f(\pi_{\theta_i})$ when a new policy π_{θ_i} crosses the threshold of its cell e. The hyperparameter $0 \leq \alpha \leq 1$, referred to as the *archive learning rate*, controls how much time is spent optimizing a region of the archive before exploring a new region. Soft archives have many theoretical and practical benefits discussed in prior work (Fontaine & Nikolaidis, 2023). Our proposed PPGA algorithm builds directly on CMA-MAEGA.

2.4 Quality Diversity Reinforcement Learning

Unlike the standard DQD formulation in which the analytical gradients of f and $\mathbf{m}$ can be computed, the QD-RL setting considers MDPs in which these functions are non-differentiable and must be *approximated* with model-free RL. The gradient approximations of f and $\mathbf{m}$ can be used to improve the performance and diversity of agents in an archive. QD-RL methods can be roughly divided into two subgroups. The first set of approaches directly optimizes over the entire archive by sampling existing policies in the archive and applying operations to the policies' parameters that either improve their performance or diversity. For example, PGA-ME (Nilsson & Cully, 2021) collects experience from evaluated agents into a replay buffer and uses TD3 to derive a policy gradient that improves the performance of randomly sampled agents from the archive, while using genetic variation (Vassiliades & Mouret, 2018) on the same set of agents to improve diversity and fill new, unexplored cells. Similarly, QDPG (Pierrot et al., 2022) derives a policy and diversity gradient using TD3 and applies these operators to randomly sampled agents in the archive.

Whereas the first family of QD-RL algorithms simultaneously search the behavioral embedding in many different regions at once, the second family uses the DQD formulation i.e., maintains a *single search policy* that explores new local regions one at a time using objective-measure gradient *approximations*. In prior work (Tjanaka et al., 2022b), the authors considered objective gradient approximations via TD3 and OpenAI-ES, while approximating the measure function gradients with

OpenAI-ES. In this work, we notice the unique on-policy nature of DQD algorithms and present a novel formulation that exploits their synergy with PPO.

3 Proposed Method: The Proximal Policy Gradient Arborescence Algorithm

We begin with the DQD algorithm CMA-MAEGA as our foundation. The algorithm can be roughly divided into three phases: (1) computing the objective-measure gradients for the branching phase, (2) providing the relative ranking of each branched policy w.r.t. the QD objective to CMA-ES, and (3) stepping the search policy in the direction of greatest archive improvement. Sections 3.1 and 3.2 focus on enabling RL optimization for phase one, 3.3 explores the connection between CMA-ES and NES and how this can improve training stability in phase two, and section 3.4 describes our method for walking the search policy with PPO in phase three.

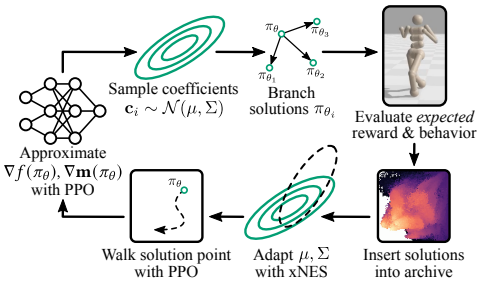


Figure 2: PPGA estimates $\nabla f, \nabla \mathbf{m}$ with PPO. We randomly sample gradient coefficients $\mathbf{c}$ and perform weighted linear recombination of the objective-measure gradients with $\mathbf{c}$ as the weights. This produces a population of gradients that, in turn, result in a population of branched policies. The policies are evaluated and inserted into the archive. xNES adapts the gradient coefficient distribution based on these insertions towards maximal archive improvement. The new mean of the coefficient distribution is used to walk the search policy towards a new, potentially unexplored region of the archive.

3.1 Markovian Measure Proxies

QD problems often contain non-Markovian measure functions. For robot locomotion tasks, the standard measure function is the proportional foot contact time with the ground $m_i(\theta) = \frac{1}{T}\sum_{t=0}^{T}\delta_i(s_t)$ for each leg $i, i = 1...k$, where the Kronecker delta $\delta_i(s_t)$ indicates whether the i'th leg is in contact with the ground or not in state s_t, and T is the episode length. However, this measure function is defined on a trajectory (i.e., the whole episode), making it non-Markovian and thus preventing us from using RL to estimate its gradient. To solve this issue, we introduce the notion of a **Markovian Measure Proxy** (MMP), which is a surrogate function that obeys the Markov property and has a positive correlation with the original measure function. For locomotion tasks, we can construct an MMP by simply removing the dependency on the trajectory and making the original measure function state-dependent, i.e., setting it to be $\delta_i(s_t)$. We can then use the exact same MDP as the standard RL formulation and replace the reward function with $\delta_i(s_t)$.

3.2 Policy Gradients for Differentiable Quality Diversity Optimization

PPO is an attractive choice as our objective-measure gradient estimator because of its ability to scale with additional parallel environments. Being an approximate trust region method, the constrained policy update step provides some robustness to noisy and non-stationary objectives. This is particularly important in the QD-RL setting, where the QD-objective is highly non-stationary – that is, the QD-objective changes with the state of the archive, which is updated on each QD iteration.

We treat the RL objective and k MMPs, each one optimized by an actor-critic pair, as reward functions to optimize. Rather than spawning a new PPO instance with a separate actor and critic network for the RL objective f and each MMP $\delta_i(s_t)$ independently, we start with a single actor $\pi_{\theta_\mu}(a|s)$ parameterized by the policy parameters θ_μ of the current search policy, and $k+1$ value functions $V_{\phi_f}, V_{\phi_{\delta_1}}, ..., V_{\phi_{\delta_k}}$. The actor is replicated $k+1$ times, each one paired with a corresponding value function. The actors are combined into a single *vectorized policy* $\pi_{<\theta_\mu^1,...,\theta_\mu^{k+1}>}(a|s)$ that jointly optimizes $< f, \delta_1(s_t), ..., \delta_1(s_t) >$ for N_1 iterations, where N_1 is a configurable hyperparameter. We additionally modify the computation of the policy gradient into a *batched policy gradient* method, where intermediate gradient estimates of each function w.r.t. policy params only flow back to

the parameters corresponding to respective individual policies during minibatch gradient descent. After N_1 iterations, we separate the vectorized policy, giving a set of subpolicies with optimized parameters $\pi_{\theta_f}(a|s), ..., \pi_{\theta_{\delta_k}}(a|s)$ that perform better w.r.t. their objectives. In the case of measure functions where $m_i(\cdot)$ is the proportion foot contact time of the $i'th$ leg, $m_i(\pi_{\theta_{\delta_k(s_t)}}) > m_i(\pi_{\theta_\mu})$ i.e. the resulting policy will have a higher proportion foot contact time over the starting policy after optimization. Subtracting the initial parameters (θ_μ) from each resulting policy gives us the desired objective-measure Jacobian $\left[\frac{\partial f}{\partial \theta_\mu}, \frac{\partial \delta_1}{\partial \theta_\mu}, ..., \frac{\partial \delta_k}{\partial \theta_\mu}\right]$, which can be linearly recombined in various ways to branch policies from θ_μ.

In addition to the VPPO implementation, we introduce the option to make the learnable action standard deviation parameter static. In the typical case, PPO decays this parameter over time in order to converge to a quasi-deterministic optimal policy at the expense of further exploration. In some environments, narrowing the action distribution can indeed help promote consistent optimal performance. In other environments, this effect can hinder the QD algorithm's ability to branch policies into new cells, given that the outer QD optimization loop relies on gradient estimates produced by PPO to discover unexplored regions of the archive. In environments where we observe this negative effect, we disable gradient flow to the action standard deviation parameter.

Finally, in order to address environmental uncertainty, we insert new policies based on their performance and behavior *averaged* over 10 parallel environments. We leverage GPU acceleration to quickly batch process many parallel environments over a population of branched policies.

3.3 Connection to Natural Evolution Strategies

We replace CMA-ES with a Natural Evolution Strategy (NES) to increase the stability and performance of CMA-MAEGA on noisy RL environments. CMA-based variants of PPGA diverged during training. Prior work (Müller & Glasmachers, 2018) showed that CMA-ES struggled to evolve deep neural network controllers with dimensionality $\mathbb{R}^d$ on stochastic RL environments. However, CMA-MAEGA uses CMA-ES to maintain search distribution in objective-measure gradient coefficient space $\mathbb{R}^{k+1} << \mathbb{R}^d$, where $k+1$ can be as small as three dimensions, implying that CMA-ES should still be effective in this low-dimensional space. It was then puzzling to find consistent divergence during the training of our CMA-based algorithm. We hypothesize that the culprit is the cumulative step-size adaptation (CSA) mechanism employed by CMA-ES. CMA-ES uses evolution paths $\rho_\sigma^{(g)}$ to adapt the step size $\sigma^{(g)}$ between successive generations (g). The mechanisms by which $\sigma^{(g)}$ are updated assume a fairly non-noisy and stationary objective f. However, the application of CMA-ES to QD optimization on stochastic RL environments presumes the exact opposite. That is, the RL objective f_{RL} is very noisy, and the QD-objective $f_{QD} = g(f_{RL}(\cdot))$, which is a function of the RL objective, is highly non-stationary, since the state of the archive $\mathcal{A}$ *changes* the direction of greatest archive improvement on every iteration. To address the training divergence, we propose using exponential evolution strategies (xNES) Glasmachers et al. (2010), a more recent and theoretically well-motivated method, as a drop in replacement for CMA-ES. Prior works have shown strong links between xNES and CMA-ES, and generalize both methods as instances of *natural evolution strategies* (Akimoto et al., 2010; Glasmachers et al., 2010). In fact, the update step in xNES is equivalent to CMA-ES up to the use of evolution paths. We refer to these prior works for an in-depth comparison. More generally, we believe *any* natural evolution strategy can be used to maintain and update the search distribution over gradient coefficients in this and any prior CMA-based DQD method.

3.4 Walking the Search Policy

In standard DQD, ∇_{step} is computed via weighted linear recombination to produce a gradient vector that steps the search policy in the least explored direction of the archive. However, the resulting gradient vector is a linearized approximation around the current search policy θ_μ and thus cannot be reused to take multiple gradient steps in a non-convex optimization problem. It would be remiss not to leverage the highly-parallelized VPPO implementation to "walk" the search policy over *many* steps in the direction of greatest archive improvement. We make the key observation that the mean gradient coefficient vector $\mathbf{c}_\mu$ of the *updated* search distribution maintained by xNES points in the direction of greatest archive improvement for the next iteration of the QD algorithm. Thus, we construct a new multi-objective reward function for VPPO to optimize by taking the dot product between the gradient

coefficient vector and the objective and measure proxies $< c_{\mu_0}, ..., c_{\mu_{k+1}} > \cdot < f, \delta_1, ..., \delta_k >$. Optimizing this function with VPPO allows us to walk the search policy θ_μ in the direction of greatest archive improvement by iteratively taking conservative steps, where the magnitude of the movement is controllable by hyperparameter N_2. This objective is stationary for all N_2 steps, and is only updated after the subsequent QD iteration. We provide pseudocode in Appendix A.

4 Experiments

We evaluate our algorithm on four different continuous-control locomotion tasks derived from the original Mujoco environments (Todorov et al., 2012): Ant, Walker2d, Half-Cheetah, and Humanoid. The standard objective in each task is to maximize forward progress and robot stability while minimizing energy consumption. We use the Brax simulator to leverage GPU acceleration and massive parallelization of the environments. The observation space sizes for these environments are 87, 17, 18, and 227, respectively, and the action space sizes are 8, 6, 6, and 17, respectively. The standard Brax environments are augmented with wrappers that determine the measures of an agent in any given rollout as implemented in QDax (Lim et al., 2022), where the number of measures of an agent is equivalent to the number of legs. The measure function is the number of times a leg contacts the ground divided by the length of the trajectory. We implement PPGA in pyribs (Tjanaka et al., 2023), with our VPPO implementation based on CleanRL's implementation of PPO (Huang et al., 2022). Most experiments were run on a SLURM cluster where each job had access to an NVIDIA RTX 2080Ti GPUs, 4 cores from a Intel(R) Xeon(R) Gold 6154 3.00GHz CPU, and 108GB of RAM. Some additional experiments and ablations were run on local workstations with access to an NVIDIA RTX 3090, AMD Ryzen 7900x 12 core CPU, and 64GB of RAM.

4.1 Comparisons

We compare our results to current state-of-the-art QD-RL algorithms: Policy Gradient Assisted MAP-Elites (PGA-ME).[1] Quality Diversity Policy Gradient (QDPG) implemented in QDax (Lim et al., 2022), and CMA-MAEGA(TD3, ES) implemented in pyribs (Tjanaka et al., 2023). We also compare against the state-of-the-art ES-based QD-RL algorithm, separable CMA-MAE (sep-CMA-MAE) (Tjanaka et al., 2022a), which allows evolutionary QD techniques to scale up to larger neural networks. Finally, in order to verify our hypothesis on the emergent synergy between PPO and DQD, we provide an ablation where TD3 is used as a drop-in replacement for PPO in PPGA, which we will refer to as TD3GA going forward. Details on the TD3GA design choices and additional ablations, such as comparing against standard PPO, can be found in the appendix. The same archive resolutions and network architectures are used for all baselines. A full list of shared hyperparameters is in Appendix B. We use an archive learning rate of 0.1, 0.15, 0.1, and 1.0 on Humanoid, Walker2d, Ant, and Half-Cheetah, respectively. Adaptive standard deviation is enabled for Ant and Humanoid. We reset the action distribution standard deviation to 1.0 on each iteration in all other environments.

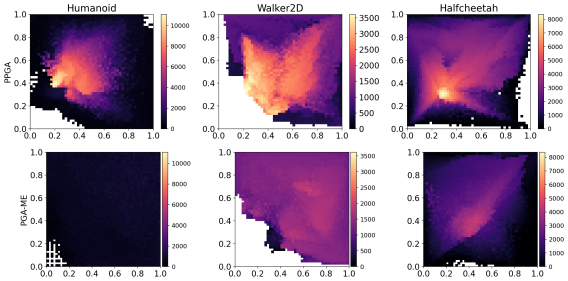


Figure 3: 2D Archive visualizations of PPGA compared to the current state-of-the-art QD-RL algorithm PGA-ME. We use 50x50 archives to show detail.

[1] A comparison on Humanoid to PBT-ME (SAC), a recent QD-RL method, can be found in Appendix H. PBT-ME (SAC) was trained with Google TPUs. Due to computational constraints, we were only able to provide a comparison on one task.

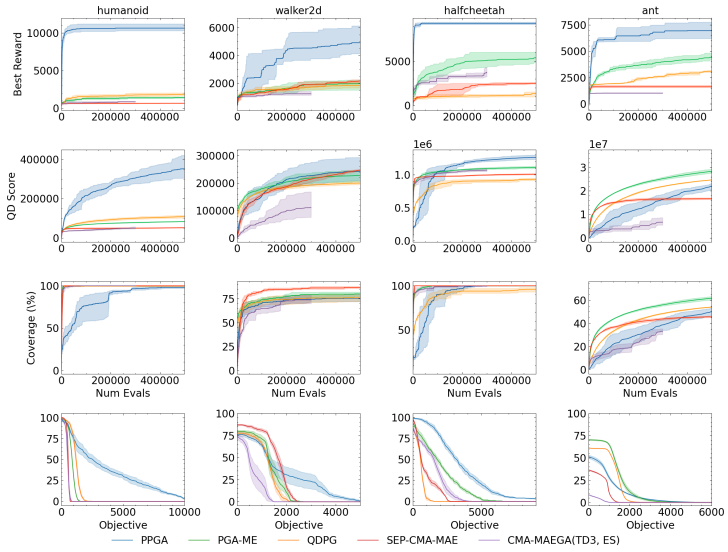


Figure 4: QD metrics and cumulative distributions for archives of PPGA and the baselines. The CCDF plots in the last row indicate the percentage of archive policies above a certain objective threshold. All plots show the mean over four seeds with a 95% bootstrapped confidence interval.

We conduct our experiments using the following criteria: **QD-score**, which is the sum of scores of all nonempty cells in the archive, and **coverage**, which is the percentage of nonempty cells in the archive, have been historically used by QD algorithms to measure performance and diversity respectively, and so we include them as metrics. However, these metrics have a number of edge cases that make them imperfect measures of performance and diversity. For example, an algorithm that fills 100% of the archive with low-performing policies can have a higher QD-score and coverage than a QD algorithm that fills fewer cells with high-performing policies. To more accurately represent the performance and diversity of a given algorithm, we additionally include plots of the **Complementary Cumulative Distribution Function** (CCDF), originally presented in (Vassiliades et al., 2016), which shows what percentage of policies in the archive achieve a reward of R or greater for all possible values of R on the x-axis. The CCDF attempts to capture notions of quality of policies in the archive and diversity, while also shedding light on how the policies are *distributed* w.r.t. performance. Finally, we include the **best reward** metric, denoting the highest-performing policy the algorithm was able to discover.

Figures 3 and 4 show that PPGA outperforms baselines in best reward and QD-score, achieving comparable coverage scores on all tasks except Ant, and generating much more illuminated archive heatmaps with a diverse range of higher performing policies than the current state of the art, PGA-ME. Notably, PPGA is the only algorithm that solves Humanoid, achieving over 4x improvement in best-performing policy and QD score compared to baselines. More important than QD-Score and Coverage are the CCDF plots. At $x = 0$, all policies in the archive are included, i.e., $x = 0$ encapsulates the coverage score. CCDF plots provide a better representation of "quality" than QD-score, since we can see how the policies in the archive are distributed. Except for Ant, PPGA produces distributions where more of the mass is distributed to the right where the high-performing policies lie.

In Figure 5, we find evidence that PPO has an important synergy with DQD that is perhaps missing in other RL algorithms. TD3GA fails to find high performing policies on Humanoid. Achieving 100% coverage is indicative of the step size σ in xNES exploding and producing highly stochastic policies that, by chance, land in far away cells. This typically occurs when xNES cannot fit a covariance matrix to the data, which in this case are weighted linear combinations of $\nabla f, \nabla \mathbf{m}$ produced by TD3.

4.2 Post-Hoc Archive Analysis

QD algorithms are known to struggle with reproducing performance and behavior in stochastic environments. To determine the replicability of our agents, we follow the guidelines in Flageat et al.

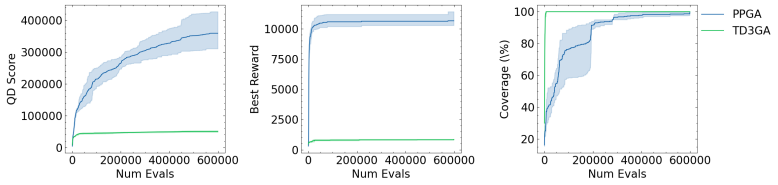
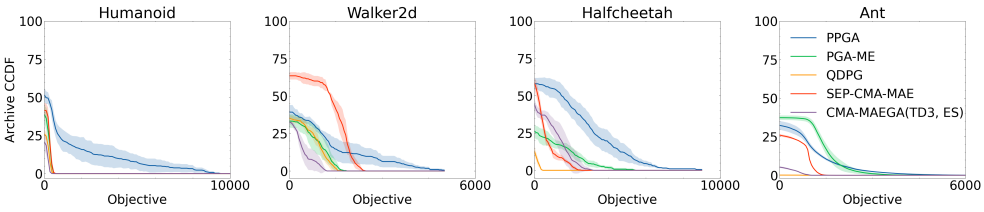


Figure 5: PPGA vs TD3GA on Humanoid on the standard QD metrics. All plots are averaged over 4 seeds. The shaded regions are the 95% bootstrapped confidence intervals.



	Humanoid			Walker2d			Halfcheetah			Ant		
	QD-Score	Cov	Best	QD-Score	Cov	Best	QD-Score	Cov	Best	QD-Score	Cov	Best
PPGA	$\mathbf{1.02 \times 10^5}$	**0.52**	**8324**	1.06×10^5	0.39	**4702**	$\mathbf{7.26 \times 10^5}$	0.58	**8919**	1.53×10^7	0.34	**7328**
PGA-ME	1.08×10^4	0.39	446	6.53×10^4	0.38	1621	3.31×10^5	0.31	4644	$\mathbf{1.79 \times 10^7}$	**0.37**	4571
sep-CMA-MAE	1.27×10^4	0.42	498	$\mathbf{1.84 \times 10^5}$	**0.64**	2326	6.03×10^5	**0.61**	228	1.16×10^7	0.30	1629
QDPG	6.53×10^3	0.26	412	6.59×10^4	0.35	1490	2.53×10^5	0.28	359	3.03×10^6	0.25	78
CMA-MAEGA (TD3, ES)	4.10×10^3	0.21	352	3.48×10^4	0.33	1025	5.70×10^5	0.56	2438	3.08×10^6	0.12	1020

Figure 6: Corrected CCDFs and Corrected QD metrics: QD-Score, *Cov*erage, *Best* Reward. Results are averaged over four seeds with error bars showing a 95% bootstrapped confidence interval.

(2023). We re-evaluate each agent in the archive 50 times and average its performance and measures to construct a Corrected Archive and use this to produce Corrected QD metrics such as QD-Score and Coverage. Fig. 6 shows the corrected QD metrics and the corrected CCDFs, respectively. After re-evaluation, PPGA maintains the lead in best reward on all tasks, QD-score on Humanoid and Ant, and Coverage on Humanoid. The CCDF plots of the Corrected Archives show PPGA producing better distributions of policies on all tasks but Ant, suggesting PPGA's policies are robust to stochasticity.

5 Discussion and Limitations

We present a new method, PPGA, which is one of the first QD-RL methods to leverage on-policy RL, the first to solve the challenging Humanoid task, and the first to achieve equivalent performance in best reward compared to standard RL on all domains. We show that DQD algorithms and on-policy RL have emergent synergies that make them work particularly well with each other. However, instead of simply combining DQD and on-policy RL as is, we re-examine the fundamental assumptions and mechanisms of each component and implement changes that maximize their synergies. There are some caveats with this approach. On-policy RL algorithms such as PPO are quite sample-inefficient and require many parallel environments per agent in order to compute the stochastic policy gradient. Although GPU acceleration and massive parallelism improve wall-clock convergence over off-policy RL, this makes our approach less sample-efficient than other off-policy QD-RL methods. Secondly, enabling PPO's adaptive standard deviation parameter (which is true by default for PPO) can have detrimental effects on PPGA's exploration capabilities, as made evident by the coverage score on Ant. This is mainly due to the fact that PPO favors collapsing the standard deviation to achieve higher average returns. In the future, we will investigate modifying the standard deviation parameter such that it dynamically shrinks or increases the standard deviation value based on the QD-optimization landscape as opposed to the RL one. Finally, we are interested to see how this method scales to even more data-rich regimes such as distributed settings, as well as its application to harder problems such as real robotics tasks. We leave these as potential avenues of future research.

6 REPRODUCIBILITY

In the supplemental material, we provide the source code and training scripts used to produce our results. In the README, we include documentation for setting up a Conda environment, running our training scripts, and visualizing our results. In addition, we provide pre-trained archives whose results were presented in this work. Detailed pseudocode and a list of relevant hyperparameters can be found in Appendices A and B.

REFERENCES

Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Bidirectional relation between cma evolution strategies and natural evolution strategies. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph (eds.), *Parallel Problem Solving from Nature, PPSN XI*, pp. 154–163, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15844-5.

Sumeet Batra, Zhehui Huang, Aleksei Petrenko, Tushar Kumar, Artem Molchanov, and Gaurav S. Sukhatme. Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 576–586. PMLR, 2021. URL https://proceedings.mlr.press/v164/batra22a.html.

Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. *Quality-Diversity Optimization: A Novel Branch of Stochastic Optimization*, pp. 109–135. Springer International Publishing, Cham, 2021. ISBN 978-3-030-66515-9. doi: 10.1007/978-3-030-66515-9_4. URL https://doi.org/10.1007/978-3-030-66515-9_4.

Tae Jong Choi and Julian Togelius. Self-referential quality diversity through differential map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, pp. 502–509, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383509. doi: 10.1145/3449639.3459383. URL https://doi.org/10.1145/3449639.3459383.

Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. Scaling map-elites to deep neuroevolution. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, pp. 67–75, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371285. doi: 10.1145/3377930.3390217. URL https://doi.org/10.1145/3377930.3390217.

Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 5027–5038. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7750-improving-exploration-in-evolution-strategies-for-deep-reinforcement-learning.pdf.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nat.*, 521(7553):503–507, 2015. doi: 10.1038/nature14422. URL https://doi.org/10.1038/nature14422.

Sam Earle, Justin Snider, Matthew C. Fontaine, Stefanos Nikolaidis, and Julian Togelius. Illuminating diverse neural cellular automata for level generation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, pp. 68–76, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392372. doi: 10.1145/3512290.3528754. URL https://doi.org/10.1145/3512290.3528754.

Manon Flageat, Felix Chalumeau, and Antoine Cully. Empirical analysis of pga-map-elites for neuroevolution in uncertain domains. *ACM Trans. Evol. Learn. Optim.*, Jan 2023. ISSN 2688-299X. doi: 10.1145/3577203. URL https://doi.org/10.1145/3577203. Just Accepted.

Matthew Fontaine and Stefanos Nikolaidis. Covariance matrix adaptation map-annealing. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pp. 456–465,

New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590389. URL https://doi.org/10.1145/3583131.3590389.

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

Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In Carlos Artemio Coello Coello (ed.), *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8-12, 2020*, pp. 94–102. ACM, 2020. doi: 10.1145/3377930.3390232. URL https://doi.org/10.1145/3377930.3390232.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *proceedings of machine learning research*, pp. 1587–1596. pmlr, 10–15 jul 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html.

Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-Efficient Design Exploration through Surrogate-Assisted Illumination. *Evolutionary Computation*, 26(3):381–410, 09 2018. ISSN 1063-6560. doi: 10.1162/evco_a_00231. URL https://doi.org/10.1162/evco_a_00231.

Tobias Glasmachers, Tom Schaul, Yi Sun, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In Martin Pelikan and Jürgen Branke (eds.), *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010*, pp. 393–400. ACM, 2010. doi: 10.1145/1830483.1830557. URL https://doi.org/10.1145/1830483.1830557.

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

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Alexander Hagg, Dominik Wilde, Alexander Asteroth, and Thomas Bäck. Designing air flow with surrogate-assisted phenotypic niching. In *International Conference on Parallel Problem Solving from Nature*, pp. 140–153. Springer, 2020.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv*, 2022.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. URL http://arxiv.org/abs/1604.00772.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius. Talakat: Bullet hell generation through constrained map-elites. In *Proceedings of The Genetic and Evolutionary Computation Conference*, pp. 1047–1054, 2018.

Joel Lehman and Kenneth O. Stanley. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223, 06 2011a. ISSN 1063-6560. doi: 10.1162/EVCO_a_00025. URL https://doi.org/10.1162/EVCO_a_00025.

Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pp. 211–218, New York, NY, USA, 2011b. Association for Computing Machinery. ISBN 9781450305570. doi: 10.1145/2001576.2001606. URL https://doi.org/10.1145/2001576.2001606.

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

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

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/28dd2c7955ce926456240b2ff0100bde-Abstract-round2.html.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/mniha16.html.

Aurélien Morel, Yakumo Kunimoto, Alex Coninx, and Stéphane Doncieux. Automatic acquisition of a repertoire of diverse grasping trajectories through behavior shaping and novelty search. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 755–761, 2022. doi: 10.1109/ICRA46639.2022.9811837.

Douglas Morrison, Peter Corke, and Jürgen Leitner. Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters*, 5(3): 4368–4375, 2020. doi: 10.1109/LRA.2020.2992195.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015. URL http://arxiv.org/abs/1504.04909.

Nils Müller and Tobias Glasmachers. Challenges in high-dimensional reinforcement learning with evolution strategies. In Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and L. Darrell Whitley (eds.), *Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part II*, volume 11102 of *Lecture Notes in Computer Science*, pp. 411–423. Springer, 2018. doi: 10.1007/978-3-319-99259-4_33. URL https://doi.org/10.1007/978-3-319-99259-4_33.

Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, pp. 866–875, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383509. doi: 10.1145/3449639.3459304. URL https://doi.org/10.1145/3449639.3459304.

Thomas Pierrot and Arthur Flajolet. Evolving populations of diverse RL agents with map-elites. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=CBfYffLqWqb.

Thomas Pierrot, Valentin Macé, Felix Chalumeau, Arthur Flajolet, Geoffrey Cideron, Karim Beguir, Antoine Cully, Olivier Sigaud, and Nicolas Perrin-Gilbert. Diversity policy gradient for sample efficient quality-diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, pp. 1075–1083, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392372. doi: 10.1145/3512290.3528845. URL https://doi.org/10.1145/3512290.3528845.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 91–100. PMLR, 2021. URL https://proceedings.mlr.press/v164/rudin22a.html.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Bryon Tjanaka, Matthew C. Fontaine, Aniruddha Kalkar, and Stefanos Nikolaidis. Training diverse high-dimensional controllers by scaling covariance matrix adaptation map-annealing, 2022a.

Bryon Tjanaka, Matthew C. Fontaine, Julian Togelius, and Stefanos Nikolaidis. Approximating gradients for differentiable quality diversity in reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, pp. 1102–1111, New York, NY, USA, 2022b. Association for Computing Machinery. ISBN 9781450392372. doi: 10.1145/3512290.3528705. URL https://doi.org/10.1145/3512290.3528705.

Bryon Tjanaka, Matthew C Fontaine, David H Lee, Yulun Zhang, Nivedit Reddy Balam, Nathaniel Dennler, Sujay S Garlanka, Nikitas Dimitri Klapsis, and Stefanos Nikolaidis. Pyribs: A bare-bones python library for quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pp. 220–229, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590374. URL https://doi.org/10.1145/3583131.3590374.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Vassilis Vassiliades and Jean-Baptiste Mouret. Discovering the elite hypervolume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pp. 149–156, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356183. doi: 10.1145/3205455.3205602. URL https://doi.org/10.1145/3205455.3205602.

Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation*, 22:623–630, 2016. URL https://api.semanticscholar.org/CorpusID:23453919.

A PPGA PSEUDOCODE

Algorithm 1 Proximal Policy Gradient Arborescence

Input: Initial policy θ_0, VPPO instance to approximate $\nabla f, \nabla \mathbf{m}$ and move the search policy, number of QD iterations N_Q, number of VPPO iterations to estimate the objective-measure functions and gradients N_1, number of VPPO iterations to move the search policy N_2, branching population size λ, and an initial step size for xNES σ_g

Initialize the search policy $\theta_\mu = \theta_0$. Initialize NES parameters $\mu, \Sigma = \sigma_g I$
for iter $\leftarrow 1$ **to** N **do**
  $f, \nabla f, \mathbf{m}, \nabla \mathbf{m} \leftarrow VPPO.compute\_jacobian(\theta_\mu, f(\cdot), \mathbf{m}(\cdot), N_1)$
  $\nabla f \leftarrow$ normalize$(\nabla f), \nabla \boldsymbol{m} \leftarrow$ normalize$(\nabla \boldsymbol{m})$
  _ $\leftarrow$ update _archive$(\theta_\mu, f, \boldsymbol{m})$
  for $i \leftarrow 1$ **to** λ **do**
    $c \sim \mathcal{N}(\mu, \Sigma)$ // sample gradient coefficients
    $\nabla_i \leftarrow c_0 \nabla f + \sum_{j=1}^{k} c_j \nabla m_j$
    $\theta_i^{'} \leftarrow \theta_\mu + \nabla_i$
    $f', *, m', * \leftarrow$ rollout(θ_i')
    $\Delta_i \leftarrow$ update_archive$(\theta_i^{'}, f', \boldsymbol{m'})$
  end for
  rank gradient coefficients ∇_i by archive improvement Δ_i
  Adapt xNES parameters $\mu = \mu', \Sigma = \Sigma'$ based on improvement ranking Δ_i
  $f'(\theta_\mu) = c_{\mu,0} f + \sum_{j=1}^{k} c_{\mu,j} m_j$, where $\mathbf{c}_\mu = \mu'$ // construct multi-objective reward function
  $\theta'_\mu = VPPO.train(\theta_\mu, f', N_2)$ // standard PPO training procedure
  if *there is no change in the archive* **then**
    Restart xNES with $\mu = 0, \Sigma = \sigma_g I$
    Set θ_μ to a randomly selected existing cell θ_i from the archive
  end if
end for

Algorithm 2 Update Archive

Input: Solution θ to insert, episodic reward f, measures $\mathbf{m} =< m_1, ..., m_k >$, archive $\mathcal{A}$, archive learning rate α
θ_{inc}, f_{inc} = $\mathcal{A}$[**m**] if $\mathcal{A}$[**m**] is nonempty else $None, 0$ // incumbent policy
$\Delta_i = 0$
if $f > f_{inc}$ **then**
  insert θ into cell $\mathcal{A}$[**m**]
  $f_{inc} \leftarrow (1 - \alpha) f_{inc} + \alpha f$
  $\Delta_i = f - f_{inc}$
end if
return Δ_i

Algorithm 3 Vectorized-PPO (VPPO)

Input: Initial search policy π_{θ_i}, objective functions to optimize $\mathbf{f} = f_1(\cdot), ..., f_k(\cdot)$, number of VPPO iterations N, number of parallel environments E, rollout length L

Initialize the vectorized agent $\vec{\pi}_{\theta_\mathrm{i}}$ = vectorized_agent($[\pi_\theta] \times (k+1)$)
for iter $\leftarrow 1$ **to** N_1 **do**
  (S, A, R, S') $\leftarrow$ rollout(vectorized_agent, E, L, **f**) // Note that **S** = $\{S_1, ..., S_k\}$, etc
  advantage $\mathbb{A}$, returns $\mathcal{G} \leftarrow$ batch_calculate_rewards(**S, A, R, S'**, **f**)
  $\vec{\pi}_\theta{}' \leftarrow$ batch_gradient_descent($\mathbb{A}, \mathcal{G}, \vec{\pi}_\theta$) // using the stochastic policy gradient
  $\vec{\pi}_\theta \leftarrow \vec{\pi}_\theta{}'$
end for
$\nabla\mathbf{f} \leftarrow \vec{\pi}_\theta{}' - \vec{\pi}_{\theta_\mathrm{i}}$
return $\nabla\mathbf{f}$

B Hyperparameters

Table 1: List of relevant hyperparameters for PPGA shared across all environments.

Hyperparameter	Value
Actor Network	[128, 128, Action Dim]
Critic Network	[256, 256, 1]
N_1	10
N_2	10
PPO Num Minibatches	8
PPO Num Epochs	4
Observation Normalization	True
Reward Normalization	True
Rollout Length	128

C Ablation Against CMA-MAEGA

PPGA makes two key changes compared to standard DQD algorithms such as CMA-MAEGA: Walking the search policy with VPPO vs. weighted linear recombination of the gradients and replacing xNES. We compare walking the search policy with VPPO to using gradient recombination in the original formulation of CMA-MEGA. In addition, we ran an ablation using xNES as the outer-loop optimizer compared to CMA-ES. However, the step-size adaptation parameter quickly diverges with CMA-ES and destabilizes training, and thus were unable to provide plots for this ablation. Fig. 7 shows the comparison of walking the search policy with VPPO versus using weighted linear recombination of the gradients. We believe that the large gap in performance is due to the fact that we can take multiple steps with VPPO via the N_2 hyperparameter. Weighted recombination results in a single gradient step where the updated search policy may land too close to the previous search policy's cell. When we branch from the updated search policy, many-branched agents will fall into overlapping cells, resulting in small archive improvement, which can lead to the emitter prematurely leaving a high-performing region of the search space.

D TD3GA Implementation Details

As part of our ablation study, we implemented TD3GA, which differs from PPGA by replacing all PPO mechanisms with TD3 (Fujimoto et al., 2018). This implementation required several algorithmic decisions, as PPGA was originally designed to integrate with an on-policy method like PPO rather than an off-policy method like TD3, In this section, we describe these decisions. In general, we intend our decisions to make TD3GA operate as closely as possible to PPGA, and we leave it to future work to explore whether variations of these decisions will further improve performance.

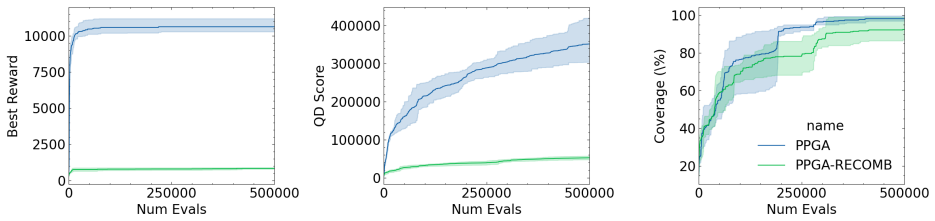


Figure 7: Ablation of walking the search policy with VPPO vs gradient recombination done in CMA-MEGA/CMA-MAEGA on the Humanoid environment.

Background: TD3 is an off-policy actor-critic method designed for single-objective RL tasks. TD3 maintains an actor (i.e., a policy) that takes actions in the environment and a critic that estimates the action-value function. TD3 also maintains a replay buffer that stores experiences collected by the actor. Over time, the actor is trained to optimize the critic. Simultaneously, based on experience in the replay buffer, the critic learns to better predict the action-value function.

Design Decisions:

1. **Number of critics:** In TD3GA, we maintain a TD3 critic for the objective function and one for each of the measure functions. We also create a separate critic for the weighted objective that is used when walking the search policy. We refer to these critics as the *objective* critic, *measure* critics, and *walking* critic.
2. **Choice of actor for critic training:** When training the critic, TD3 requires an actor that generates actions for states sampled from the replay buffer. Prior QD-RL methods that estimate objective gradients with TD3, e.g., PGA-ME (Nilsson & Cully, 2021) and CMA-MEGA (TD3, ES) (Tjanaka et al., 2022b), fulfill this role with a dedicated actor. This actor is referred to as a *greedy actor* since its primary purpose is to optimize its performance with respect to the critic.

   In theory, since TD3 is an off-policy method, any actor, including a greedy actor, can be used to train the critic. However, to make TD3GA closer to PPGA, we instead use a copy of the current search policy to train the critic. This decision provides our TD3 instances with an on-policy nature that mirrors the PPO mechanisms found in PPGA.
3. **When to train critics:** Similar to the PPO value functions in PPGA, we maintain all critics throughout the entire training run, updating them on every iteration.
4. **Experience collection:** This decision concerns which experience collected in the environment should be inserted into the replay buffer. With the settings in our paper, PPGA (and TD3GA) samples 300 policies per iteration, and each policy is evaluated for 10 episodes, with each episode having up to 1,000 timesteps of experience; in total, these policies generate 3 million timesteps per iteration. The typical replay buffer size (Fujimoto et al., 2018) in TD3 is 1 million, meaning the buffer would be filled three times over if we inserted all of this experience, i.e., the critic would never be trained with the experience from two-thirds of all policies.

   To ensure that we collect experience from many policies across many iterations, we make two decisions. First, we only collect one episode of experience from each agent — this already cuts down experience collected on each iteration to 300,000 timesteps. Second, we increase the replay buffer size to 5 million to store experience across more iterations.

   Note that there is no analog to the replay buffer in PPGA since PPO is an on-policy method. Instead, PPO regresses the value function based on experience collected while evaluating the policy.
5. **Gradient computation:** We begin by describing how a prior method estimates the objective gradient with TD3. To estimate the objective gradient of a policy, CMA-MEGA (TD3, ES) samples a batch of experience and passes the batch through the corresponding actor. The actions outputted by the actor are then inputted to the critic. Backgpropagating through the critic and the actor then provides the objective gradient.

Roughly, the above procedure corresponds to taking a *single* step of TD3. However, in PPGA, the gradient is computed by taking the difference after *multiple* steps of PPO (the number of steps is determined by the hyperparameters N_1 and N_2. The straightforward approach to mirror this behavior in TD3GA is to also output a gradient after several steps of TD3. Thus, when training each critic, we also track the final state of the actor (recall that the actor used in critic training is a copy of the search policy). At the end of critic training, our gradient is then the difference between the final actor and the original search policy.

For example, to compute the objective gradient, we train the objective critic with an actor that is a copy of the search policy. While training the critic, the actor is updated so that it maximizes the objective critic. After N_1 steps, we compute the objective gradient as the difference between the actor and the original search policy.

6. **Actor target networks:** One TD3 mechanism that improved stability and performance was target networks, which are slowly updating versions of the actor and critic parameters. In TD3GA, it is straightforward to apply this mechanism to the critic parameters.

   However, since the actor is reset to the current search policy on every iteration, it is difficult to maintain a single target network. Thus, on every iteration, the target network for the actor is simply reset to the current search policy's parameters before the gradient computation.

Pseudocode: Algorithm 4 shows the process for computing an objective gradient in TD3GA. The same process applies to computing measure gradients. The process for walking the search policy is also similar, except that the reward is a weighted combination of the objective and measures (the weights come from the mean μ of the emitter's coefficient distribution). Furthermore, when walking the search policy, we return the final actor instead of a gradient.

Hyperparameters: TD3GA inherits all relevant hyperparameters from PPGA (Table 1). It also inherits TD3 hyperparameters from prior QD-RL works that incorporate TD3 (Tjanaka et al., 2022b; Nilsson & Cully, 2021), except for having a larger replay buffer. We also increase the batch size used during critic training to mimic the batch size used by PPO to regress the value function. Table 2 lists hyperparameters shared across all environments. Note that the archive learning rate depends on the environment but is identical to that used in PPGA.

Table 2: List of hyperparameters used in TD3GA, shared across all environments.

Hyperparameter	Value
Actor Network	[128, 128, Action Dim]
Critic Network	[256, 256, 1]
Training Steps for Objective and Measures (N_1)	10
Training Steps for Walking (N_2)	10
Optimizer	Adam
Adam Learning Rate	3×10^{-4}
Target Network Update Rate (τ)	0.005
Target Network Update Frequency (d)	2
Smoothing Noise Standard Deviation (σ_p)	0.2
Smoothing Noise Clip (c)	0.5
Discount Factor (γ)	0.99
Replay Buffer Size	5,000,000
Batch Size (n_{batch})	48,000

Algorithm 4 TD3 Gradient Computation for the Objective. Adapted from TD3 (Fujimoto et al., 2018)

Input: Current search policy params θ_μ, current TD3 critic networks Q_{ψ_1} and Q_{ψ_2} parameterized by ψ_1 and ψ_2 respectively, current critic targets ψ'_1 and ψ'_2, replay buffer B
Hyperparameters: Training steps N_1 or N_2, target network update rate τ, target network update frequency d, smoothing noise standard deviation σ_p, smoothing noise clip c, discount factor γ, batch size n_{batch}

Initialize actor $\phi = \theta_\mu$, actor target $\phi' = \theta_\mu$

{*Either N_1 for gradient computation or N_2 for walking the search policy*}
for iter $\leftarrow 1$ **to** N_1 **do**
  {*r is replaced with the measures for measure gradients, or a weighted combination of the reward and measures for walking the search policy*}
  Sample mini-batch of n_{batch} transitions (s, a, r, s') from B

  {*Train the critics*}
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_p), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\psi'_i}(s', \tilde{a})$
  {*This update is performed with Adam*}
  Update critics $\psi_i \leftarrow \arg\min_{\psi_i} \frac{1}{N} \sum (y - Q_{\psi_i}(s, a))^2$

  if $t \mod d$ **then**
    {*Update ϕ by the deterministic policy gradient with Adam*}
    $\nabla_\phi J(\phi) = \frac{1}{n_{batch}} \sum \nabla_a Q_{\psi_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    {*Update target networks:*}
    $\psi'_i \leftarrow \tau\psi_i + (1 - \tau)\psi'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

{*Note that ψ_1, ψ_2, ψ'_1, and ψ'_2 are maintained across calls to this function*}

{*When walking the search policy, we just return the new policy params ϕ*}
$\nabla f \leftarrow \phi - \theta_\mu$
return ∇f

E HYPERPARAMETER STUDY

We investigate the effects of changing the N_1 and N_2 hyperparameters on the performance of PPGA in the Humanoid domain. We run 4 seeds of PPGA on the following combinations of N_1 and N_2: (10, 5), (5, 10), and (1, 1), and compare against the baseline (10, 10). With these experiments, we wish to address the following questions

1. How few PPO steps on both the objective-measure Jacobian calculation and walking the current search policy can we take before noticing performance degradation?
2. How do asymmetric choices for N_1 and N_2 (i.e. more Jacobian calculation steps than walking steps and vice versa) affect performance?

(5, 10) achieves the most consistent results while achieving the same performance as the baseline, while (10, 5) results in very high run-to-run variance and lower coverage. This suggests that spending more computation on walking the current search policy is more important than approximating the objective-measure Jacobian. Nonetheless, we conclude that in a more computationally constrained setting, PPGA could be tuned to perform fewer N_1 and N_2 steps while still maintaining good performance. Finally, (1, 1) performs the worst, achieving slightly more than 50% of the baseline's performance, implying that there is significant performance degradation when too few Jacobian-calculation and walking steps are taken.

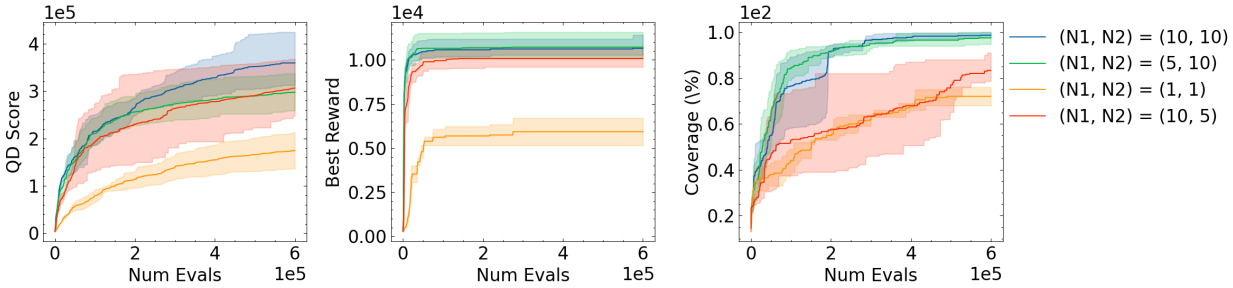


Figure 8: Study of the effect of different hyperparameter choices for N_1 and N_2. N_1 is the number of PPO steps used to calculate the objective-measure Jacobian, and N_2 is the number of PPO steps used to walk the search policy. All plots are averaged over 4 seeds. The shaded region is the 95% boostrapped confidence interval.

F PPO Ablations

We perform ablations against standard PPO on Humanoid, the most challenging of all domains, in order to compare PPGA's relative performance against standard RL. We first ablate how the measure functions affect performance. The algorithm "PPGA (No Measure)" performs exactly as standard PPGA but with no measure functions. In this case, PPGA computes gradients for the RL objective when branching and deciding where in the archive to move next. Unsurprisingly, PPGA (No Measures) achieves the same best reward as PPGA, but with less than half the archive coverage.

In the second ablation, we run vanilla PPO and store the intermediate policies in between mini-batch gradient descent steps in an archive. The algorithm, dubbed "PPO + Archive", achieves the same best reward as PPGA, but with less than 1% archive coverage. Note that PPGA (No Measures) is still performing an outer-loop optimization step of the QD-objective, thus achieving better coverage than PPO + Archive, whereas PPO + Archive only optimizes for the RL objective.

Algorithm	**QD-Score**	**Coverage**	**Best Reward**
PPGA	3.59×10^5	98.67%	9677
PPGA (No Measures)	8.24×10^4	32.06%	9653
PPO + Archive	3.30×10^3	0.08%	9651

Table 3: Ablation study of PPGA with various components on Humanoid. PPGA (No Measures) functions exactly as PPGA, but only computes objective gradients and walks the search policy with respect to f. PPO + Archive runs standard PPO and stores the intermediate policy updates as policies into an archive. Archives are 10x10.

G SCALING EXPERIMENTS

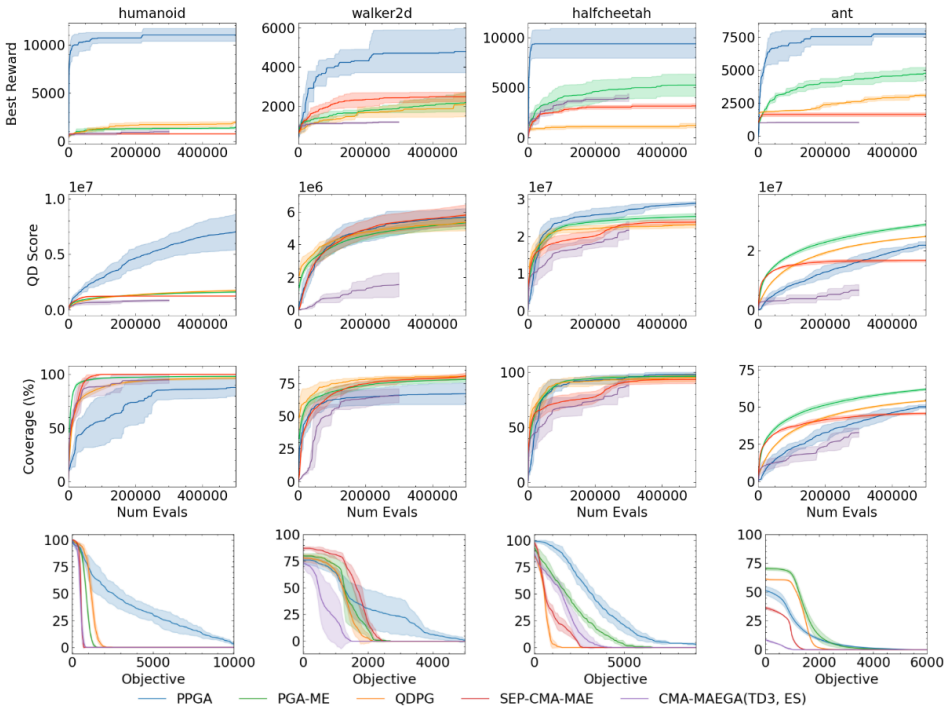


Figure 9: QD-Score, Coverage, Best Reward, and CCDF plots for PPGA and baselines, with 50x50 archives for all tasks except for Ant. Ant retains the same 10^4 archive resolution, as this is already sufficiently large.

We test the scalability of PPGA along two axes – the ability to scale to larger archives and the ability to learn with more data. To test for the first property, we scale up the archive resolution to 50x50 for locomotion tasks with two measures i.e. Humanoid, Walker2D, and Halfcheetah and compare against baselines. PPGA retains the same performance across tasks, with slight reductions in coverage. This is due to PPGA being an entirely gradient-based method, using gradients for branching and walking the search policy, whereas prior methods that employ ES can get lucky and randomly mutate policies into far away cells.

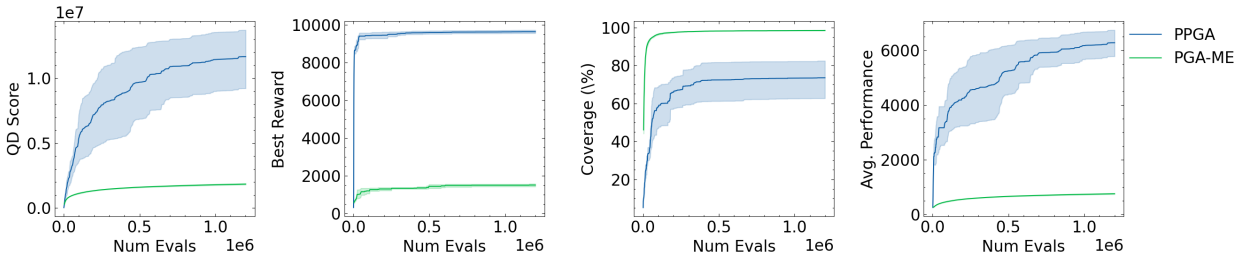


Figure 10: QD Metrics for 50x50 archives of PPGA and PGA-ME trained to 1.2 million evaluations. Results are averaged over 4 seeds. The additional "average performance" metric is presented to show how performance over all policies in the archive changes with additional training.

Finally, we run our algorithm on a 50x50 archive with 1.2 million evaluations, more than twice as long as the main experiments, of both PPGA and the state of the art baseline PGA-ME, on Humanoid. We report an additional metric, the average performance of the archive, which is the sum of scores of all policies divided by the total number of policies in the archive. This can also be interpreted as the normalized QD score. We find that PPGA continues to improve on this metric, indicating that the episodic returns and thus performance of many policies in the archive continues to improve with additional training. We observe this effect to a much lesser extent with PGA-ME.

H Comparing to PBT-ME (SAC)

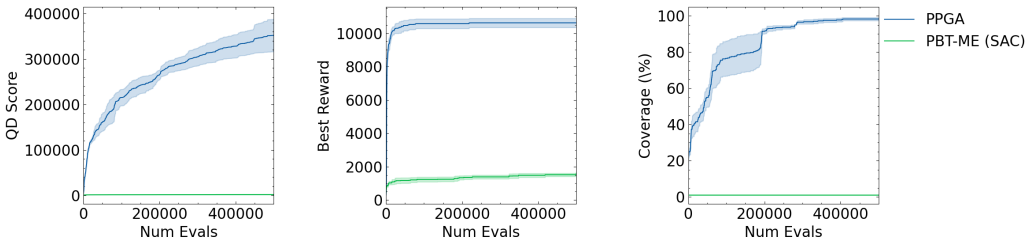


Figure 11: Comparison to PBT-ME (SAC) on Humanoid, which uses Soft Actor-Critic (Haarnoja et al., 2018) to compute the policy gradient when improving the agents.

Population-Based Training MAP-Elites (PBT-ME) (Pierrot & Flajolet, 2023) is a recent QD-RL algorithm that alleviates hyperparameter sensitivity in QD-RL algorithms by evolving populations of agents and their hyperparameters, while also using the policy gradient formulation to improve the agents' performance. The evolved and optimized policies are added to an archive following the MAP-Elites formulation. PBT-ME was run with Google TPUs – due to computational constraints, we were only able to make comparisons against PBT-ME on the Humanoid domain, which we present here. Specifically, we compare to the PBT-ME (SAC) variant.