Scaling Covariance Matrix Adaptation MAP-Annealing to High-Dimensional Controllers

Bryon Tjanaka, Matthew C. Fontaine, Aniruddha Kalkar, Stefanos Nikolaidis
University of Southern California, United States
{tjanaka, mfontain, kalkar, nikolaid}@usc.edu

Abstract

Pre-training a diverse set of robot controllers in simulation has enabled robots to adapt online to damage in robot locomotion tasks. However, finding diverse, high-performing controllers requires specialized hardware and extensive tuning of a large number of hyperparameters. On the other hand, the Covariance Matrix Adaptation MAP-Annealing algorithm, an evolution strategies (ES)-based quality diversity algorithm, does not have these limitations and has been shown to achieve state-of-the-art performance in standard benchmark domains. However, CMA-MAE cannot scale to modern neural network controllers due to its quadratic complexity. We leverage efficient approximation methods in ES to propose three new CMA-MAE variants that scale to very high dimensions. Our experiments show that the variants outperform ES-based baselines in benchmark robotic locomotion tasks, while being comparable with state-of-the-art deep reinforcement learning-based quality diversity algorithms. Source code and videos are available at `https://scalingcmamae.github.io`

1 Introduction

By generating a diverse collection of controllers, we can endow a robot with a wide variety of useful behaviors, such as moving an arm into different configurations [1, 2], walking in every direction [3, 4], pushing an object through many trajectories [5], or traveling to any position in a maze [6, 7, 8]. Having a collection of controllers has proven particularly useful in robotic locomotion, where the collection enables a walking robot to adapt to damage [9, 10, 11]. The controllers differ by how long each foot contacts the ground, such that if one foot is damaged, the robot can select a controller that does not rely on that foot.

Searching for these diverse controllers may be viewed as a quality diversity (QD) optimization problem [6]. In QD, the goal is to find solutions ϕ that are diverse with respect to one or more measure functions $\boldsymbol{m}(\phi)$ while maximizing an objective function $f(\phi)$. In robotic locomotion, we search for controller policies π_ϕ parameterized by ϕ. Each controller should satisfy a unique output of the measure function by using its feet in a different manner from the other controllers, while optimizing the objective by walking forward quickly.

To solve a QD problem, a QD algorithm must balance two aspects given a limited compute budget: *exploring* the measure space and *optimizing* the objective. Furthermore, one of these aspects may be more important for a given problem. For example, in locomotion, exploring corresponds to finding new controllers which use the robot's feet a different amount, while optimizing corresponds to making existing controllers walk faster. Prior work [12] shows that optimization is more difficult than exploration in this domain, as it is trivial to explore the measure space by standing in place and lifting the feet up and down to achieve a new style of foot usage.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

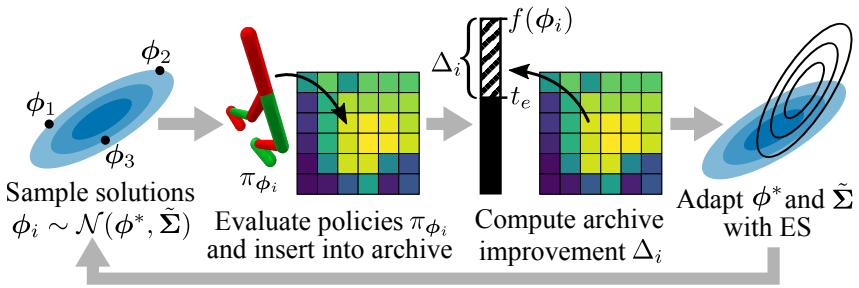


Figure 1: We propose variants of the CMA-MAE algorithm which scale to high-dimensional controllers. The variants maintain a Gaussian search distribution with mean ϕ^* and approximate covariance matrix $\tilde{\Sigma}$. Solutions ϕ_i sampled from the Gaussian are evaluated and inserted into an archive, where they generate improvement feedback Δ_i based on their objective value $f(\phi_i)$ and a threshold t_e which each archive cell maintains. Finally, the Gaussian is updated with an evolution strategy (ES). Our variants differ from CMA-MAE by selecting scalable ESs, as the CMA-ES used in CMA-MAE has $\Theta(n^2)$ time complexity per sampled solution.

Prior algorithms [13, 12] seem to strike a balance between these two aspects of QD in locomotion tasks, leading to state-of-the-art results. However, these algorithms have practical limitations due to their dependence on deep reinforcement learning (RL) methods. Namely, they require a GPU, must perform time-consuming training of a neural network, and have a large number of hyperparameters.

Recent work [14] suggests evolution strategies (ES) as a compelling alternative to deep RL methods when optimizing a single controller. Compared with deep RL, ESs can run entirely on CPU and do not require network training, and ESs such as CMA-ES [15] are designed to have almost no hyperparameters. Given these benefits, prior work [12, 11] has developed QD algorithms based on ESs, but these methods have not yet been able to match the performance of deep RL-based QD methods.

On the other hand, the recently proposed ES-based Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) algorithm [16] has proven adept at trading off between the exploration and objective optimization aspects of QD. Through a single hyperparameter $\alpha \in [0, 1]$, CMA-MAE can be tuned to optimize only the objective (as in CMA-ES), or prioritize moving away from previously discovered high-quality solutions (as in CMA-ME [17]), and anywhere in between. Selecting any value of α strictly between 0 and 1 yields theoretical benefits that result in state-of-the-art performance on several QD benchmark domains. We hypothesize that because CMA-MAE balances this tradeoff, the algorithm would excel in robotic locomotion tasks with hard-to-optimize objectives.

While CMA-MAE excels at moderate dimensional domains, the algorithm is computationally intractable for robotic locomotion due to the high dimensionality of modern neural network controllers. CMA-MAE relies on CMA-ES [15], whose $\Theta(n^2)$ time complexity per sampled solution, where n is the number of parameters, makes it impossible for CMA-MAE to optimize the thousands or even millions of parameters found in such controllers.

CMA-ES's complexity arises from how it models the search distribution by maintaining a Gaussian with a full rank $n \times n$ covariance matrix. However, a full rank covariance matrix is often not necessary, so prior work [18, 19, 20, 21, 22, 23] scales CMA-ES to higher dimensions by replacing the covariance matrix with sparse approximations. These CMA-ES variants scale to solve a variety of high-dimensional optimization problems.

Thus, *our key insight is that we can scale CMA-MAE to high-dimensional controllers by adopting the same approximations in its CMA-ES components*. Though our CMA-MAE variants leverage sparse covariance matrix approximations, we hypothesize these variants will be competitive.

We scale CMA-MAE by proposing three CMA-MAE variants, which we evaluate on robotic locomotion tasks from QDGym [24]. We show that our variants are the highest-performing QD methods based solely on ES. Furthermore, they inherit the aforementioned practical benefits of ES, and they are comparable to the state-of-the-art deep RL-based QD method PGA-ME [13] on three of four tasks. We are excited about future applications in other domains, such as robotic manipulation [25] and scenario generation [26].

2 Problem Statement

Quality diversity (QD). Drawing from recent work [27], we define QD as considering an objective function $f(\boldsymbol{\phi})$ and k-dimensional measure function $\boldsymbol{m}(\boldsymbol{\phi})$,[1] where $\boldsymbol{\phi} \in \mathbb{R}^n$ is an n-dimensional solution. The outputs of $\boldsymbol{m}$ form a k-dimensional *measure space* $\mathcal{X}$. The QD objective is to find solutions $\boldsymbol{\phi}$ such that $\forall \boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{m}(\boldsymbol{\phi}) = \boldsymbol{x}$ and $f(\boldsymbol{\phi})$ is maximized. Solving this QD objective would require infinite memory because $\mathcal{X}$ is a continuous space, so algorithms based on MAP-Elites [9, 10] relax the QD objective by discretizing $\mathcal{X}$ into a tesselation $\mathcal{Y}$ of M cells. Then, the QD objective is to find an *archive* (i.e., a collection) $\mathcal{A}$ containing M elites $\boldsymbol{\phi}_i$, where each $\boldsymbol{\phi}_i$ has measures $\boldsymbol{m}(\boldsymbol{\phi}_i)$ corresponding to a cell in $\mathcal{Y}$ and $f(\boldsymbol{\phi}_i)$ is maximized. We express this relaxed QD objective as $\max_{\boldsymbol{\phi}_{1..M}} \sum_{i=1}^{M} f(\boldsymbol{\phi}_i)$.

Quality diversity for reinforcement learning (QD-RL). As defined in prior work [12], QD-RL is a special instance of QD in which $\boldsymbol{\phi}$ parameterizes a reinforcement learning (RL) agent's policy $\pi_{\boldsymbol{\phi}}$ and the objective is the *expected discounted return* of the agent. QD-RL extends Markov Decision Processes (MDPs) [28] and is formulated as a tuple $(\mathcal{S}, \mathcal{U}, p, r, \gamma, \boldsymbol{m})$. Here, on each timestep of an episode, an agent observes state $s \in \mathcal{S}$ and takes action $a \in \mathcal{U}$ with policy $\pi_{\boldsymbol{\phi}}(a|s)$. The agent transitions to the next state $s' \in \mathcal{S}$ according to $s' \sim p(\cdot|s, a)$ and receives reward $r(s, a, s')$. The *expected discounted return* is defined as $f(\boldsymbol{\phi}) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, s'_t)\right]$ for a discount factor $\gamma \in (0, 1)$ and episode length T. Finally, QD-RL includes a k-dimensional measure function $\boldsymbol{m}(\boldsymbol{\phi})$ which describes the agent's behavior during an episode.

3 Background

3.1 MAP-Elites

Many QD algorithms, including the ones in this paper, are based on the MAP-Elites algorithm. The original MAP-Elites algorithm [9, 10] divides the measure space equally along each dimension, resulting in an archive consisting of grid cells. Initially, MAP-Elites samples solutions from a predefined distribution and inserts them into the archive by mapping each solution to a cell based on its measures. On subsequent iterations, MAP-Elites samples a solution $\boldsymbol{\phi}$ from the archive, mutates it by adding Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and adds the new solution $\boldsymbol{\phi}'$ to the archive. Crucially, if a solution ever lands in the same cell as a previous solution, it is only stored in the archive if it has a higher objective. Hence, solutions in the archive are referred to as *elites*.

Extensions of MAP-Elites include new archive structures [29, 30] and integrations with surrogate models [31, 32, 33, 34, 35]. MAP-Elites has also been applied to environment generation for human-robot interaction [36, 37] and dataset generation for manipulation [38].

Policy Gradient Assisted MAP-Elites (PGA-ME) [13]. Since MAP-Elites performs poorly when directly optimizing modern RL controllers [11, 13, 12], several algorithms [11, 13, 12, 39] have scaled it to such controllers. One such algorithm, PGA-ME, replaces the Gaussian noise mutation with two types of operations: (1) gradient ascent, performed with the TD3 algorithm [40], and (2) crossover, performed with a genetic algorithm [1]. PGA-ME has demonstrated state-of-the-art performance on the robotic locomotion tasks evaluated in this paper. We include PGA-ME and MAP-Elites as experimental baselines.

3.2 Large-Scale Evolution Strategies (ES)

ESs [41] optimize continuous parameters by adapting a population of solutions, such that the population is more likely to have high performance. Here we discuss ESs designed for high-dimensional search spaces. In Sec. 4, we show how these ESs help scale CMA-MAE.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [15]. CMA-ES is an approximate second-order method which follows a natural gradient [42] and demonstrates state-of-the-art results in black-box optimization [43]. CMA-ES maintains a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ that models the distribution

[1] It is common to define $\boldsymbol{m}(\boldsymbol{\phi})$ via k separate measure functions $m_i(\boldsymbol{\phi})$. Note that in robotics settings, prior work has sometimes referred to measure function outputs as behavior descriptors or behavior characteristics.

of search directions. In every iteration, CMA-ES samples λ solutions from the Gaussian and updates it based on rankings of the solutions' performance.

CMA-ES requires $\Theta(n^2)$ space and has $\Theta(n^2)$ runtime per sampled solution due to its $n \times n$ covariance matrix $\boldsymbol{\Sigma}$. The runtime stems from two operations. First, updating $\boldsymbol{\Sigma}$ requires matrix-vector multiplications. Second, since it is easy to sample from the standard Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on a computer, sampling from the Gaussian is implemented as

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \sim \boldsymbol{\mu} + \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \sim \boldsymbol{\mu} + \boldsymbol{\Sigma}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{1}$$

Computing the *transformation matrix* $\boldsymbol{\Sigma}^{\frac{1}{2}}$ requires an $O(n^3)$ eigendecomposition, which CMA-ES amortizes to $O(n^2)$ per sampled solution by only recomputing $\boldsymbol{\Sigma}^{\frac{1}{2}}$ every $\frac{n}{\lambda}$ iterations. Nevertheless, this complexity is intractable for high n. To scale CMA-ES, prior works [18, 19, 20, 21, 22, 23] assume that a full Gaussian is unnecessary for modeling the search distribution in high dimensions and thus replace the covariance matrix with a more efficient approximation. We discuss two such approaches below.

Separable CMA-ES (sep-CMA-ES) [18]. As the simplest large-scale CMA-ES variant, sep-CMA-ES assumes that the problem is separable, i.e., that it may be solved by optimizing each parameter individually. Hence, sep-CMA-ES constrains the covariance matrix to be diagonal, so it only stores n parameters. Updates and sampling then require $\Theta(n)$ runtime per solution, as the matrix-vector multiplications are replaced with dot products and $\boldsymbol{\Sigma}^{\frac{1}{2}}$ in Eq. 1 is computed by taking the square root of the diagonal elements. sep-CMA-ES outperforms CMA-ES on separable benchmarks and even on some non-separable problems.

Limited-Memory Matrix Adaptation ES (LM-MA-ES) [19]. LM-MA-ES is another large-scale CMA-ES variant. It maintains a rank-k approximation of the transformation matrix $\boldsymbol{\Sigma}^{\frac{1}{2}}$. This approximation consists of k vectors which represent different directions of the search distribution. Since the approximation is rank-k, the space complexity is $\Theta(kn)$, and the time complexity is $\Theta(kn)$ per solution. Overall, LM-MA-ES performs well on optimization benchmarks, matching other large-scale versions of CMA-ES.

OpenAI-ES [14]. Unlike the algorithms above, OpenAI-ES is not a CMA-ES variant. Instead, inspired by Natural Evolution Strategies [44], OpenAI-ES maintains an isotropic Gaussian with constant covariance matrix $\sigma \boldsymbol{I}$, and it only updates the mean $\boldsymbol{\phi}^*$ via gradient ascent with Adam [45]. The gradient for this update is computed based on the rank-normalized [46, 44] performances of λ solutions sampled from the Gaussian with mirror sampling [47]. OpenAI-ES has performed well across RL benchmarks [14, 48, 49]. It has a time complexity of $\Theta(n)$ per sampled solution.

3.3 Combining ES with MAP-Elites

Here we discuss algorithms which enhance MAP-Elites by integrating ESs.

MAP-Elites with Evolution Strategies (ME-ES) [11]. ME-ES maintains a solution point $\boldsymbol{\phi}^*$ and modifies it with OpenAI-ES in two phases: *exploit*, where OpenAI-ES optimizes $\boldsymbol{\phi}^*$ to have high objective value, and *explore*, where OpenAI-ES optimizes $\boldsymbol{\phi}^*$ to be novel [7, 50] (far away from previous solutions in measure space). Both phases begin by sampling a new $\boldsymbol{\phi}^*$ from the archive. They then run for k iterations, with only $\boldsymbol{\phi}^*$ inserted into the archive after each iteration. ME-ES is a baseline in our experiments.

Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) [16]. One promising class of QD algorithms combines CMA-ES [15] with MAP-Elites. We directly extend the most recent such algorithm, CMA-MAE, which has shown state-of-the-art performance in QD benchmark domains. CMA-MAE maintains an archive which records a *threshold* t_e for each cell e. t_e is initialized to a user-defined minimum objective min_f. When a solution $\boldsymbol{\phi}$ is inserted into the archive, it is placed into its corresponding cell e if its objective value $f(\boldsymbol{\phi})$ exceeds t_e. Then, t_e is updated via polyak averaging, i.e. $t_e \leftarrow (1-\alpha)t_e + \alpha f(\boldsymbol{\phi})$, where $\alpha \in [0, 1]$ is the *archive learning rate*. Note that the solution's objective value only needs to cross the threshold, rather than the objective value of the elite previously in the cell. Hence, the archive does not always store the best solutions like MAP-Elites does, so implementations must track these best solutions separately.

For generating solutions, CMA-MAE queries a set of *emitters* on each iteration. Each emitter maintains its own Gaussian search distribution with mean $\boldsymbol{\phi}^*$ and covariance matrix $\boldsymbol{\Sigma}$. In every

iteration, the emitter samples solutions from this Gaussian and inserts the solutions into the archive. When a solution $\boldsymbol{\phi}$ is inserted, the archive returns an *improvement value* $\Delta_i \leftarrow f(\boldsymbol{\phi}) - t_e$, i.e., the difference between the objective value of the solution and its corresponding cell's threshold. The emitter ranks its generated solutions by these improvement values. Based on this ranking, the emitter updates $\boldsymbol{\phi}^*$ and $\boldsymbol{\Sigma}$ with CMA-ES, such that it is more likely to generate solutions with high improvement in the future.

A key parameter in CMA-MAE is the archive learning rate α. When $\alpha = 0$, the threshold always remains at min_f, and hence the improvement is always equivalent to the objective value (minus the constant min_f). This makes CMA-MAE equivalent to CMA-ES, as it optimizes solely for the objective function. However, when $\alpha = 1$, the threshold is always equivalent to the objective value of the elite currently in the cell, which means there is minimal improvement for inserting a solution into that cell. In this case, CMA-MAE is equivalent to the previously developed CMA-ME [17], where discovering new cells in the archive (i.e., exploration) always takes priority over improving the objective value of cells where there is already a solution. Hence, by varying α from 0 to 1, we trade off between optimizing the objective and moving away from previously discovered elites. In particular, we use a low α value due to the need for optimization in robotic locomotion domains.

Covariance Matrix Adaptation MAP-Elites via a Gradient Arborescence (CMA-MEGA) [27]. In the differentiable quality diversity (DQD) setting, where exact objective and measure gradients are available, CMA-ME and gradient ascent have been combined to form the state-of-the-art CMA-MEGA algorithm. To extend CMA-MEGA to QD-RL, prior work replaces the exact gradients which CMA-MEGA relies on with gradient approximations, since the RL environment is usually non-differentiable. This results in two variants: CMA-MEGA (ES) approximates all gradients with OpenAI-ES [14], while CMA-MEGA (TD3, ES) approximates the objective gradient with TD3 [40] and measure gradients with OpenAI-ES. These variants are baselines in our experiments.

4 Scaling Covariance Matrix Adaptation MAP-Annealing

In CMA-MAE, each emitter relies on CMA-ES to update the Gaussian search distribution from which it samples solutions. Since CMA-ES requires $\Theta(n^2)$ runtime per solution (where n is the solution dimension), CMA-MAE is intractable for neural network controllers, where there are thousands to millions of parameters. To scale CMA-MAE, we propose three variants which substitute CMA-ES with large-scale ESs. These variants differ primarily in the complexity of their covariance matrix approximation:

- **sep-CMA-MAE** substitutes sep-CMA-ES [18], which constrains the covariance matrix $\boldsymbol{\Sigma}$ to be diagonal, resulting in $\Theta(n)$ complexity.
- **LM-MA-MAE** substitutes LM-MA-ES [19], which approximates the transformation matrix $\boldsymbol{\Sigma}^{\frac{1}{2}}$ with $k \ll n$ n-dimensional vectors, leading to $\Theta(kn)$ complexity.
- **OpenAI-MAE** substitutes OpenAI-ES [14], which sets the covariance matrix $\boldsymbol{\Sigma}$ to be the identity matrix. Though $\boldsymbol{\Sigma}$ is constant, various operations still necessitate $\Theta(n)$ complexity.

The listed complexities refer to (1) the space required per emitter, as each emitter maintains its own ES instance, and (2) the runtime required per sampled solution, which is the same regardless of the number of emitters.

Algorithm 1 shows the pseudocode for these CMA-MAE variants, and Fig. 1 shows an overview of the variants. Each variant proceeds as follows. First, it initializes the archive along with each emitter's ES parameters (line 2-3). This step includes initializing the covariance matrix approximation $\tilde{\boldsymbol{\Sigma}}$ in lieu of CMA-MAE's full covariance matrix $\boldsymbol{\Sigma}$. Next, the variant repeatedly queries the emitters for solutions (line 4). Each emitter (line 5) samples λ solutions from the distribution $\mathcal{N}(\boldsymbol{\phi}^*, \tilde{\boldsymbol{\Sigma}})$ (line 6-7). Note that the sampling procedure depends on the approximation employed by the variant. Once sampled, the solutions are evaluated and inserted into the archive if they cross their cell's threshold t_e (line 8-15). Then, the ES's parameters (including $\tilde{\boldsymbol{\Sigma}}$) are updated based on the improvement ranking of the solutions (line 16-17), such that the emitter is more likely to sample solutions with high improvement on the next iteration. Finally, the emitter is restarted if the ES converges (line 18-20). We adopt the default update and convergence rules from each ES.

Algorithm 1: LM-MA-MAE, sep-CMA-MAE, and OpenAI-MAE. Adapted from CMA-MAE [16]. Highlighted lines show differences from CMA-MAE.

CMA-MAE variants $(eval, \boldsymbol{\phi}_0, N, \psi, \lambda, \sigma, \alpha, min_f)$**:**
- **Input:** $eval$ function which rolls out policy $\pi_{\boldsymbol{\phi}}$ and outputs objective $f(\boldsymbol{\phi})$ and measures $\boldsymbol{m}(\boldsymbol{\phi})$, initial solution $\boldsymbol{\phi}_0$, iterations N, number of emitters ψ, batch size λ, initial step size σ, archive learning rate α, minimum objective min_f
- **Result:** Generates $N\psi\lambda$ solutions, storing elites in an archive $\mathcal{A}$
- Initialize archive $\mathcal{A}$ and threshold $t_e \leftarrow min_f$ for every cell e
- Initialize ψ emitters, each with mean $\boldsymbol{\phi}^* \leftarrow \boldsymbol{\phi}_0$, covariance matrix approximation $\tilde{\boldsymbol{\Sigma}} \leftarrow \sigma \boldsymbol{I}$, and internal parameters $\boldsymbol{p}$
- **for** $iter \leftarrow 1..N$ **do**
  - **for** *Emitter* 1 .. *Emitter* ψ **do**
    - **for** $i \leftarrow 1..\lambda$ **do**
      - $\boldsymbol{\phi}_i \sim \mathcal{N}(\boldsymbol{\phi}^*, \tilde{\boldsymbol{\Sigma}})$
      - $f(\boldsymbol{\phi}_i), \boldsymbol{m}(\boldsymbol{\phi}_i) \leftarrow eval(\boldsymbol{\phi}_i)$
      - $e \leftarrow$ calculate_cell($\mathcal{A}, \boldsymbol{m}$)
      - $\Delta_i \leftarrow f(\boldsymbol{\phi}_i) - t_e$
      - **if** $f(\boldsymbol{\phi}_i) > t_e$ **then**
        - Replace the solution in cell e of archive $\mathcal{A}$ with $\boldsymbol{\phi}_i$
        - $t_e \leftarrow (1-\alpha)t_e + \alpha f(\boldsymbol{\phi}_i)$
      - **end**
    - **end**
    - Rank $\boldsymbol{\phi}_i$ by Δ_i
    - Adapt $\boldsymbol{\phi}^*, \tilde{\boldsymbol{\Sigma}}, \boldsymbol{p}$ based on improvement ranking Δ_i
    - **if** *ES algorithm converges* **then**
      - Restart emitter with $\boldsymbol{\phi}^* \leftarrow$ a randomly selected elite in $\mathcal{A}$, $\tilde{\boldsymbol{\Sigma}} \leftarrow \sigma \boldsymbol{I}$, and new internal parameters $\boldsymbol{p}$
    - **end**
  - **end**
- **end**

5 Experiments

To test our CMA-MAE variants' ability to create diverse, high-performing locomotion controllers, we evaluate it on four robotic locomotion tasks from the QDGym [24] benchmark based on PyBullet [51, 52] and OpenAI Gym [53]. In each environment (Fig. 2), the objective is to walk forward quickly, while the measures track the proportion of time in an episode that each of the robot's feet touches the ground, i.e., if a robot has four legs, it has four measures.

As prior work [12] notes, these environments are challenging because they require focusing on objective optimization. It is easy to fill the archive with poor controllers that achieve different measures by standing still and lifting the feet up and down, but making these controllers walk forward is much more difficult. We compare our variants with five baselines: PGA-ME, two CMA-MEGA variants (CMA-MEGA (ES) and CMA-MEGA (TD3, ES)), ME-ES, and MAP-Elites.[2]

5.1 Hyperparameters

As in prior work [13, 12], the robot controller is a neural network mapping states to actions. The network has two hidden layers of size 128 and tanh activations on every layer but the input, and it is initialized with Xavier initialization. The number of parameters in each domain is as follows: QD Ant (21,256), QD Half-Cheetah (20,742), QD Hopper (18,947), QD Walker (20,230).

[2]We consider algorithms in the MAP-Elites family which address the QD-RL problem formulated in Sec. 2 without making any assumptions on the measures, as opposed to algorithms based on novelty search [54] or algorithms which define measures similarly to an RL reward function [39].

Figure 2: QDGym locomotion environments [24]. Images reproduced with permission from prior work [12].

All algorithms are allocated 1 million evaluations, with each solution evaluated for one episode. PGA-ME, CMA-MEGA (ES), CMA-MEGA (TD3, ES), and ME-ES use the same hyperparameters as in their prior works [13, 12, 11], except ME-ES uses 200 samples. MAP-Elites uses a noise standard deviation of $\sigma = 0.02$ and a batch size of 100.

All CMA-MAE variants run with a low archive learning rate $\alpha = 0.001$ to encourage optimization of the objective and with $\psi = 5$ emitters, each with batch size $\lambda = 40$ and initial step size $\sigma = 0.02$. LM-MA-MAE sets $k = \lambda = 40$. The Adam optimizer for OpenAI-ES in OpenAI-MAE uses learning rate 0.01 and L2 regularization coefficient 0.005.

For min_f, QDGym does not have predefined minimum objectives, but we adopt values from prior work [12] which recorded the minimum objective inserted into an archive during their experiments: QD Ant (-374.70), QD Half-Cheetah (-2,797.52), QD Hopper (-362.09), QD Walker (-67.17).

5.2 Experimental Design

We conduct a between-groups study with the algorithm (LM-MA-MAE, sep-CMA-MAE, OpenAI-MAE, PGA-ME, CMA-MEGA (ES), CMA-MEGA (TD3, ES), ME-ES, MAP-Elites) and environment (QD Ant, QD Half-Cheetah, QD Hopper, QD Walker) as independent variables and QD score as dependent variable. The QD score [6] measures performance holistically by summing the objectives of all elites in the archive. To ensure that no elite subtracts from the QD score (objectives may be negative), we subtract min_f from all elites' objectives before computing the QD score. We repeat our experiments 5 times to test three hypotheses:

H1: *All CMA-MAE variants will outperform prior ES-based methods, i.e., ME-ES and CMA-MEGA (ES).*

H2: *All CMA-MAE variants will outperform deep RL-based methods, i.e., PGA-ME and CMA-MEGA (TD3, ES).*

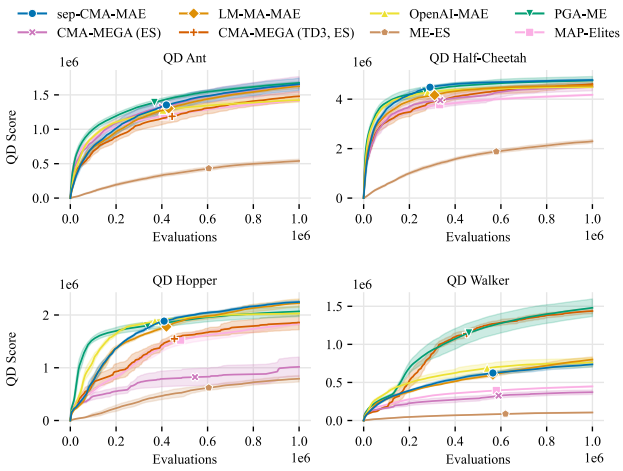
H3: *The performances of the CMA-MAE variants will be ordered as LM-MA-MAE $>$ sep-CMA-MAE $>$ OpenAI-MAE.*

H1 and H2 are based on the key difference between the variants and the baselines, i.e., the smooth improvement ranking based on archive thresholds. Since CMA-MEGA (ES) and CMA-MEGA (TD3, ES) use a standard MAP-Elites archive (equivalent to setting $\alpha = 1$ in CMA-MAE), we think they will focus too much on exploration and be unable to optimize existing solutions. Meanwhile, PGA-ME and ME-ES separate measure space exploration from objective optimization; PGA-ME has two types of operations, while ME-ES has two distinct phases. We hypothesize that this separation will not work as well as blending the two aspects together via the improvement ranking.

The ordering in H3 is based on the complexity of each variant's covariance matrix approximation. Essentially, given the difficulty of the locomotion domains, we hypothesize that a better representation of the search distribution will lead to higher performance.

5.3 Implementation

Each algorithm runs with 100 CPUs on a high-performance cluster. Algorithms which train TD3 are allocated one NVIDIA Tesla P100 GPU. We obtained data for all baselines with consent from the authors of prior work [12]. All algorithms are implemented with the pyribs library [55].



	QD Ant			QD Half-Cheetah		
	QD	Cov	Best	QD	Cov	Best
sep-CMA-MAE	1.655	0.81	2,309.14	**4.773**	**1.00**	2,398.57
LM-MA-MAE	1.624	0.82	2,336.64	4.545	**1.00**	2,219.24
OpenAI-MAE	1.424	0.81	2,135.07	4.490	**1.00**	2,244.54
PGA-ME	**1.674**	0.96	**2,843.86**	4.759	**1.00**	**2,746.98**
CMA-MEGA (ES)	1.650	0.96	2,213.06	4.489	**1.00**	2,265.73
CMA-MEGA (TD3, ES)	1.480	0.97	2,482.83	4.613	**1.00**	2,486.10
ME-ES	0.540	0.63	2,515.20	2.297	0.95	1,911.33
MAP-Elites	1.418	**0.98**	1,506.97	4.176	**1.00**	1,822.88

	QD Hopper			QD Walker		
	QD	Cov	Best	QD	Cov	Best
sep-CMA-MAE	**2.247**	0.95	3,018.47	0.737	0.89	1,359.52
LM-MA-MAE	2.225	0.96	**3,022.31**	0.800	0.89	1,404.56
OpenAI-MAE	2.029	0.90	2,695.97	0.761	0.91	1,400.75
PGA-ME	2.069	0.97	2,884.08	**1.480**	0.99	**2,619.17**
CMA-MEGA (ES)	1.017	0.97	1,441.00	0.372	**1.00**	940.50
CMA-MEGA (TD3, ES)	1.858	**0.98**	2,597.87	1.437	**1.00**	2,302.31
ME-ES	0.792	0.74	2,642.30	0.105	0.86	1,025.74
MAP-Elites	1.836	**0.98**	2,602.94	0.448	**1.00**	989.31

Figure 3: Experimental results. Plots show QD score of each algorithm in each QDGym environment with respect to the number of solution evaluations. The table shows QD score ("QD") as a multiple of 10^6 as well as archive coverage ("Cov", the fraction of archive cells filled with an elite) and best performance ("Best", the objective value of the highest-performing archive elite). Both the plots and the table show the mean over 5 trials. Error bars show standard error of the mean.

6 Results

Fig. 3 summarizes our results. To test our hypotheses (Sec. 5.2), we first ran a two-way ANOVA to examine the effect of the algorithm and environment on QD score. The ANOVA showed a significant interaction between algorithm and environment ($F(21, 128) = 18.01, p < 0.001$). Thus, for each hypothesis, we report the results of simple main effects analysis with Bonferroni corrections.

H1: All CMA-MAE variants outperformed ME-ES in all environments and CMA-MEGA (ES) in two environments (QD Hopper and QD Walker). In QD Ant and QD Half-Cheetah, there was no significant difference between the variants and CMA-MEGA (ES).

H2: In QD Ant, Half-Cheetah, and Hopper, the variants had no significant difference with PGA-ME and CMA-MEGA (TD3, ES), except OpenAI-MAE was significantly worse than PGA-ME in QD Ant. In QD Walker, PGA-ME and CMA-MEGA (TD3, ES) outperformed all variants.

H3: We ran pairwise comparisons between all three pairs of variants. In QD Ant, there was no significant difference between LM-MA-MAE and sep-CMA-MAE, but OpenAI-MAE was significantly worse than LM-MA-MAE. In QD Half-Cheetah, sep-CMA-MAE outperformed both OpenAI-MAE and LM-MA-MAE, but there was no significant difference between OpenAI-MAE

and LM-MA-MAE. In QD Hopper, sep-CMA-MAE outperformed OpenAI-MAE, but there was no significant difference between sep-CMA-MAE and LM-MA-MAE or between LM-MA-MAE and OpenAI-MAE. Finally, there was no significant difference between any pair of variants in QD Walker.

7 Discussion and Conclusion

Our results show that ES-based methods are a viable alternative in quality diversity for reinforcement learning (QD-RL). First, the analysis of **H1** shows that the CMA-MAE variants are the highest-performing ES-based method in QD-RL, either matching or exceeding the existing CMA-MEGA (ES) and ME-ES on all domains. Additionally, the analysis of **H2** shows that the variants are comparable to RL-based methods (including the state-of-the-art PGA-ME) in 3 of 4 tasks. While the variants do not solve QD Walker, prior work [12] highlights the importance of deep RL in this task, as only algorithms with TD3 (PGA-ME and CMA-MEGA (TD3, ES)) have performed well here. Overall, these results show that the improvement ranking and annealing mechanisms of CMA-MAE enable it to focus on the optimization of the objective, which is necessary in the robotic locomotion tasks.

We note that a benefit of ES-based methods is their generality. While PGA-ME performs well on the robotic locomotion tasks, it is designed only for QD-RL settings, since its TD3 component is applicable only in MDP domains. On the other hand, the CMA-MAE variants treat the objective as a black box and can be applied to general QD problems outside of robotics.

Furthermore, ES-based methods have attractive practical benefits. Specifically:

(1) The CMA-MAE variants are noticeably light on computation. While PGA-ME and CMA-MEGA (TD3, ES) both train a TD3 critic, a lengthy process which requires a GPU, the CMA-MAE variants run entirely on CPU and are three times faster (7 hrs vs 20 hrs in QD Ant and QD Half-Cheetah, and 4 hrs vs 13 hrs in QD Hopper and QD Walker).

(2) The CMA-MAE variants have substantially fewer hyperparameters than PGA-ME and the CMA-MEGA variants since they depend on CMA-ES and its variants, which are designed to be parameterized only by an initial step size σ and batch size λ. Hence, the CMA-MAE variants only require 5 hyperparameters ($\psi, \lambda, \sigma, \alpha, min_f$, see Algorithm 1). In contrast, PGA-ME and CMA-MEGA (TD3, ES) require a large number of parameters for TD3 and their other components.

When comparing the CMA-MAE variants in **H3**, we observe that OpenAI-MAE performs significantly worse than one other variant in three out of four tasks. This is because OpenAI-MAE approximates the covariance matrix with a constant identity matrix, as opposed to the more complex approximations of sep-CMA-MAE and LM-MA-MAE. On the other hand, there does not seem to be a significant difference between sep-CMA-MAE and LM-MA-MAE. This result warrants further investigation, and it indicates that both the rank-k and diagonal approximations of the covariance matrix are sufficient for modeling the distribution of search directions in the locomotion tasks.

In conclusion, our work scales CMA-MAE to robotic locomotion controllers by replacing the CMA-ES component of CMA-MAE with efficient approximations. Our results show the CMA-MAE variants are effective in benchmark locomotion tasks and have fewer hyperparameters than the deep RL baselines, making them attractive to practitioners. Since CMA-MAE is a general purpose algorithm, we are excited about future work that will test the CMA-MAE variants in domains beyond robot locomotion, such as robotic manipulation [25] and scenario generation [26]. Future work will also test the pretrained controllers in the real world and will explore the computational benefits of recent hardware-accelerated frameworks [56].

Acknowledgments and Disclosure of Funding

This work was partially supported by the NSF CAREER (#2145077), NSF NRI (#2024949), and NSF GRFP (#DGE-1842487).

References

[1] V. Vassiliades and J.-B. Mouret, "Discovering the elite hypervolume by leveraging interspecies correlation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, (New York, NY, USA), p. 149–156, Association for Computing Machinery, 2018.

[2] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, pp. 580–586, Feb 2021.

[3] A. Cully and J.-B. Mouret, "Evolving a Behavioral Repertoire for a Walking Robot," *Evolutionary Computation*, vol. 24, pp. 59–88, 03 2016.

[4] A. Cully, "Multi-emitter map-elites: Improving quality, diversity and data efficiency with heterogeneous sets of emitters," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, (New York, NY, USA), p. 84–92, Association for Computing Machinery, 2021.

[5] A. Cully, "Autonomous skill discovery with quality-diversity and unsupervised descriptors," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, (New York, NY, USA), p. 81–89, Association for Computing Machinery, 2019.

[6] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[7] J. Lehman and K. O. Stanley, "Abandoning Objectives: Evolution Through the Search for Novelty Alone," *Evolutionary Computation*, vol. 19, pp. 189–223, 06 2011.

[8] J. Miao, T. Zhou, K. Shao, M. Zhou, W. Zhang, J. Hao, Y. Yu, and J. Wang, "Promoting quality and diversity in population-based reinforcement learning via hierarchical trajectory space exploration," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7544–7550, 2022.

[9] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, pp. 503–507, 05 2015.

[10] J. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *CoRR*, vol. abs/1504.04909, 2015.

[11] C. Colas, V. Madhavan, J. Huizinga, and J. Clune, "Scaling map-elites to deep neuroevolution," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, (New York, NY, USA), p. 67–75, Association for Computing Machinery, 2020.

[12] B. Tjanaka, M. C. Fontaine, J. Togelius, and S. Nikolaidis, "Approximating gradients for differentiable quality diversity in reinforcement learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, (New York, NY, USA), p. 1102–1111, Association for Computing Machinery, 2022.

[13] O. Nilsson and A. Cully, "Policy gradient assisted map-elites," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, (New York, NY, USA), p. 866–875, Association for Computing Machinery, 2021.

[14] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017.

[15] N. Hansen, "The CMA evolution strategy: A tutorial," *CoRR*, vol. abs/1604.00772, 2016.

[16] M. C. Fontaine and S. Nikolaidis, "Covariance matrix adaptation map-annealing," *arXiv*, vol. abs/2205.10752, 2022.

[17] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance matrix adaptation for the rapid illumination of behavior space," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, (New York, NY, USA), p. 94–102, Association for Computing Machinery, 2020.

[18] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *Parallel Problem Solving from Nature – PPSN X* (G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, eds.), (Berlin, Heidelberg), pp. 296–305, Springer Berlin Heidelberg, 2008.

[19] I. Loshchilov, T. Glasmachers, and H. Beyer, "Large scale black-box optimization by limited-memory matrix adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 353–358, 2019.

[20] I. Loshchilov, "LM-CMA: An Alternative to L-BFGS for Large-Scale Black Box Optimization," *Evolutionary Computation*, vol. 25, pp. 143–171, 03 2017.

[21] Y. Akimoto and N. Hansen, "Online model selection for restricted covariance matrix adaptation," in *Parallel Problem Solving from Nature – PPSN XIV* (J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, eds.), (Cham), pp. 3–13, Springer International Publishing, 2016.

[22] Y. Akimoto and N. Hansen, "Projection-based restricted covariance matrix adaptation for high dimension," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, (New York, NY, USA), p. 197–204, Association for Computing Machinery, 2016.

[23] Z. Li and Q. Zhang, "A simple yet efficient evolution strategy for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 637–646, 2018.

[24] O. Nilsson, "Qdgym." `https://github.com/ollenilsson19/QDgym`, 2021.

[25] A. Morel, Y. Kunimoto, A. Coninx, and S. Doncieux, "Automatic acquisition of a repertoire of diverse grasping trajectories through behavior shaping and novelty search," *arXiv preprint arXiv:2205.08189*, 2022.

[26] M. Fontaine and S. Nikolaidis, "A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy," *Robotics: Science and Systems*, 2021.

[27] M. C. Fontaine and S. Nikolaidis, "Differentiable quality diversity," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[29] V. Vassiliades, K. Chatzilygeroudis, and J. Mouret, "Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 623–630, 2018.

[30] M. C. Fontaine, S. Lee, L. B. Soros, F. De Mesentier Silva, J. Togelius, and A. K. Hoover, "Mapping hearthstone deck spaces through map-elites with sliding boundaries," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, (New York, NY, USA), p. 161–169, Association for Computing Machinery, 2019.

[31] V. Bhatt, B. Tjanaka, M. C. Fontaine, and S. Nikolaidis, "Deep surrogate assisted generation of environments," 2022.

[32] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-Efficient Design Exploration through Surrogate-Assisted Illumination," *Evolutionary Computation*, vol. 26, pp. 381–410, 09 2018.

[33] P. Kent and J. Branke, "Bop-elites, a bayesian optimisation algorithm for quality-diversity search," 2020.

[34] B. Lim, L. Grillotti, L. Bernasconi, and A. Cully, "Dynamics-aware quality-diversity for efficient learning of skill repertoires," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 5360–5366, 2022.

[35] Y. Zhang, M. C. Fontaine, A. K. Hoover, and S. Nikolaidis, "Deep surrogate assisted map-elites for automated hearthstone deckbuilding," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, (New York, NY, USA), p. 158–167, Association for Computing Machinery, 2022.

[36] M. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the Importance of Environments in Human-Robot Coordination," in *Proceedings of Robotics: Science and Systems*, (Virtual), July 2021.

[37] M. C. Fontaine and S. Nikolaidis, "Evaluating human-robot interaction algorithms in shared autonomy via quality diversity scenario generation," *J. Hum.-Robot Interact.*, jul 2021. Just Accepted.

[38] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.

[39] T. Pierrot, V. Macé, F. Chalumeau, A. Flajolet, G. Cideron, K. Beguir, A. Cully, O. Sigaud, and N. Perrin-Gilbert, "Diversity policy gradient for sample efficient quality-diversity optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, (New York, NY, USA), p. 1075–1083, Association for Computing Machinery, 2022.

[40] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596, PMLR, 10–15 Jul 2018.

[41] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, Mar 2002.

[42] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Bidirectional relation between cma evolution strategies and natural evolution strategies," in *Parallel Problem Solving from Nature, PPSN XI* (R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, eds.), (Berlin, Heidelberg), pp. 154–163, Springer Berlin Heidelberg, 2010.

[43] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, 06 2001.

[44] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *Journal of Machine Learning Research*, vol. 15, no. 27, pp. 949–980, 2014.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[46] D. Ha, "A visual guide to evolution strategies," *blog.otoro.net*, 2017.

[47] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, and T. Hohm, "Mirrored sampling and sequential selection for evolution strategies," in *Parallel Problem Solving from Nature, PPSN XI* (R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, eds.), (Berlin, Heidelberg), pp. 11–21, Springer Berlin Heidelberg, 2010.

[48] P. Pagliuca, N. Milano, and S. Nolfi, "Efficacy of modern neuro-evolutionary strategies for continuous control optimization," *Frontiers in Robotics and AI*, vol. 7, p. 98, 2020.

[49] J. Lehman, J. Chen, J. Clune, and K. O. Stanley, "Es is more than just a traditional finite-difference approximator," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, (New York, NY, USA), p. 450–457, Association for Computing Machinery, 2018.

[50] J. Lehman and K. 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, (New York, NY, USA), p. 211–218, Association for Computing Machinery, 2011.

[51] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." `http://pybullet.org`, 2016–2020.

[52] B. Ellenberger, "Pybullet gymperium." `https://github.com/benelot/pybullet-gym`, 2018–2019.

[53] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.

[54] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 5027–5038, Curran Associates, Inc., 2018.

[55] B. Tjanaka, M. C. Fontaine, D. H. Lee, Y. Zhang, T. T. M. Vu, S. Sommerer, N. Dennler, and S. Nikolaidis, "pyribs: A bare-bones python library for quality diversity optimization." `https://github.com/icaros-usc/pyribs`, 2021.

[56] B. Lim, M. Allard, L. Grillotti, and A. Cully, "Accelerated quality-diversity for robotics through massive parallelism," *arXiv preprint arXiv:2202.01258*, 2022.