EVOLUTIONARY DIVERSITY OPTIMIZATION WITH CLUSTERING-BASED SELECTION FOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning (RL) has achieved significant successes, which aims to obtain a single policy maximizing the expected cumulative rewards for a given task. However, in many real-world scenarios, e.g., navigating in complex environments and controlling robots, one may need to find a set of policies having both high rewards and diverse behaviors, which can bring better exploration and robust few-shot adaptation. Recently, some methods have been developed by using evolutionary techniques, including iterative reproduction and selection of policies. However, due to the inefficient selection mechanisms, these methods cannot fully guarantee both high quality and diversity. In this paper, we propose EDO-CS, a new Evolutionary Diversity Optimization algorithm with Clustering-based Selection. In each iteration, the policies are divided into several clusters based on their behaviors, and a high-quality policy is selected from each cluster for reproduction. EDO-CS also adaptively balances the importance between quality and diversity in the reproduction process. Experiments on various (i.e., deceptive and multi-modal) continuous control tasks, show the superior performance of EDO-CS over previous methods, i.e., EDO-CS can achieve a set of policies with both high quality and diversity efficiently while previous methods cannot.

1 INTRODUCTION

Reinforcement Learning (RL) is an effective method for training agents to make decisions in a given environment, which is often to obtain a policy maximizing the expected cumulative rewards (Sutton & Barto, 2018; Li, 2017). As RL is used in more real-world scenarios, training a single policy is not enough to handle complex problems, and we may need to find a set of policies with both high quality and diverse behaviors. For example, when navigating in complex environments (Zhu et al., 2017; Mirowski et al., 2017), there are usually deceptive rewards trapping a single policy into the local optima. Maintaining a set of policies allows data to be collected with diverse behaviors and may help address challenging exploration problems in these environments. When controlling robots (Cully et al., 2015), it is fragile to have only one policy. In contrast, maintaining a set of policies can not only improve robustness, e.g., enable real robots to recover quickly from joint damage (Cully et al., 2015), but also be helpful for few-shot adaption when facing unseen tasks (Kumar et al., 2020).

Evolutionary Algorithms (EAs) are general-purpose heuristic optimization algorithms that maintain a population, i.e., a set of solutions, and simulate the natural evolution process with iterative reproduction and selection (Bäck, 1996). In each iteration of EAs, the parents are selected according to the selection mechanism and then used to reproduce the offspring. Quality-Diversity (QD) algorithms (Pugh et al., 2016; Chatzilygeroudis et al., 2020) are the particularly exciting class of EAs that aims to return a set of high-

quality solutions with diverse behaviors in a single run. Thus, it is a natural idea to apply QD algorithms in RL to solve the challenging problems in complex environments (Conti et al., 2018; Colas et al., 2020; Parker-Holder et al., 2020; Cideron et al., 2020).

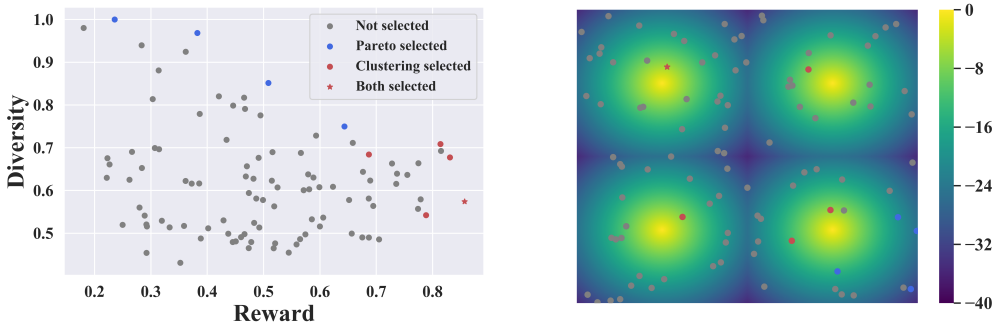
NSR-ES (Conti et al., 2018) hybrids Evolution strategies (ES) (Salimans et al., 2017) with Novelty Search (NS) (Lehman & Stanley, 2011b) to improve the performance on RL tasks with sparse or deceptive rewards. It maintains a population and adds a novelty term to the objective function of ES. In each iteration, NSR-ES selects one policy to do reproduction, where the probability of selecting a particular policy is proportional to its novelty score. ME-ES (Colas et al., 2020) also selects one policy in each iteration. Different from NSR-ES, it maintains an archive to store the policies generated before and uses a biased selection mechanism that alternatively selects the policy according to its quality and the diversity from the archive. NSR-ES and ME-ES can avoid local optima in some complex environments. However, the selection mechanisms they used, i.e., only selecting one policy in each iteration, may cause only a specific behavior to be well optimized and can not return a set of high-quality policies with diverse behaviors.

In contrast to the two algorithms mentioned above, DvD-ES (Parker-Holder et al., 2020) select a population rather than a single policy and uses a population-wide diversity criterion that maximizes the volume between the behaviors of the policies and optimizes all policies of a population simultaneously. This joint update allows DvD-ES to maintain a diverse population in various environments. However, DvD-ES always selects the offspring of the last iteration and totally ignores the previous information. This may lead to 'cycling', a phenomenon where the population keeps moving alternatively between two areas, making DvD-ES inefficient.

An alternative way is to view quality and diversity as two optimization objectives and uses Pareto-based selection, which is a typically used selection method in multi-objective optimization (Deb et al., 2002; Liu & Qian, 2021). For more information on multi-objective optimization or Pareto-based selection, please refer to Deb (2011). The Pareto-based selection is commonly used in the QD fields (Lehman & Stanley, 2011a) and has recently been introduced into RL by QD-RL (Cideron et al., 2020). In each iteration of QD-RL, the Pareto front is firstly calculated with respect to quality and diversity. Then, the population is selected from the Pareto front based on the crowding distance. The Pareto-based selection can guarantee the population is uniformly distributed on the Pareto front; however, it may not guarantee the population is uniformly distributed in the behavior space, which is actually the goal of QD algorithms. As shown in Figure 1, the policies selected by Pareto-based selection still behave similarly. To the best of our knowledge, we are the first to point out this phenomenon.

The selection mechanisms in the QD algorithms above, such as selecting all offspring, biased selection, and Pareto-based selection, all have their disadvantages, making it difficult for the algorithms to reach the goal of QD. To this end, we propose an Evolutionary Diversity Optimization algorithm with Clustering-based Selection (EDO-CS). In each iteration, policies in the archive are divided into several clusters based on their behaviors, and a high-quality policy is selected for reproduction. This selection mechanism guarantees the selected policies of high quality and distributed in different areas of the behavior space. We illustrate this by an example in Figure 1.

Figure 1(a) shows the objective space, where the two objectives are reward and diversity of the policies (denoted by colored points). Figure 1(b) shows the behavior space, where the distance of the two points implies the similarity of the behaviors of the corresponding policies. Besides, we use the heat map to show the quality of the policies, and four optima are distributed in different areas in the behavior space. A set of policies should be selected from the current archive to do reproduction. Intuitively, we should select the policies with high quality in each of the four optimal areas. Although the points selected by Pareto-based selection (blue points) are well distributed on the Pareto front in Figure 1(a), they are concentrated in the lower right corner of the behavior space, which means that they behave similarly in some sense. By contrast, the points selected by our clustering-based method (red points) are distributed in all the four optimal



(a) Objective space

(b) Behavior space

Figure 1: The selected solutions by Pareto-based selection and clustering-based selection on a synthetic problem. Figure 1(a) shows the diversity (novelty) and reward of each policy, and Figure 1(b) shows the behavior space, where the heat map shows the quality of the policy.

areas of the behavioral space, successfully obtaining a set of high-quality policies with diverse behaviors. It is because that our clustering-based selection firstly divides the policies into several clusters. Even if we randomly select one policy from each cluster, we can still get a diverse population. Then, to have a high-quality population, we just need to select a high-quality policy in each cluster.

To examine the performance of EDO-CS, we conduct experiments on a variety of continuous control tasks from *OpenAI Gym* library (Brockman et al., 2016). Firstly, we show that EDO-CS can solve navigating problems in environments with deceptive rewards. Then, we test it in multi-modal tasks, showing that EDO-CS can efficiently find optimal policies with diverse behaviors. In the standard MuJoCo environments that do not require much diversity, we show that EDO-CS can achieve even better performance than Vanilla ES that focuses on quality. We also test the influence of the hyper-parameters of EDO-CS in these environments.

The remainder of this paper is organized as follows. We first introduce some definitions and notations used in our paper in Section 2. After that, we propose our method in Section 3. The results and analysis of experiments are shown in Section 4. Finally, in Section 5, we conclude this work.

2 Background

We consider the setting of a fully observable Markov Decision Process (MDP) describing as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space (Sutton & Barto, 2018). $P(s_{t+1}|s_t, a_t)$ gives the probability of transiting to state s_{t+1} after taking action a_t at state s_t, $r_t = R(s_t, a_t, s_{t+1})$ is the reward obtained by an agent transiting to state s_{t+1} after taking action a_t at state s_t, and γ is the discount factor.

A policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ is a mapping from state space $\mathcal{S}$ to action space $\mathcal{A}$, paremeterized by θ, which can be either deterministic or randomized. The goal is to optimize parameters θ such that π_θ maximizes the expected cumulative rewards $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$, where τ is the trajectory over a horizon T. The objective can be written as follows:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \right]. \tag{1}$$

General RL algorithms use the expected cumulative rewards as objective. To escape from local optima, NS further encourages policies to seek for different behaviors from those previously seen (Lehman & Stanley, 2011b; Conti et al., 2018). In NS, the behavior characterization $b(\pi_\theta)$ is used to describe the behavior of a given policy π_θ, which is usually domain-dependent. For example, it can be the coordinates of the final location (x, y) of the robot in the robot locomotion problem, and the corresponding behavior space is $\mathbb{R}^2$. Besides, NS maintains an archive A to store the policies generated before.

Given the behavior characterization$b(\pi_\theta)$ and the archive A, the diversity of a particular policy π_θ is computed as the average Euclidean distance between $b(\pi_\theta)$ and $A'_{\pi_\theta,k}$, i.e., the set of k-nearest neighbors of policy π_θ in the archive A:

$$Div(\theta) = \frac{1}{k} \sum_{\pi'_\theta \in A'_{\pi_\theta,k}} \|b(\pi_\theta) - b(\pi'_\theta)\|_2. \tag{2}$$

Then, the objective to be maximized of a particular policy π_θ can be expressed as:

$$J(\theta) = (1-\lambda)\mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right] + \lambda Div(\theta), \tag{3}$$

where λ is the degree of seeking for diversity.

3 METHOD

In this section, we introduce the proposed algorithm, Evolutionary Diversity Optimization algorithm with Clustering-based Selection (EDO-CS). We give an overview of EDO-CS in Section 3.1. Then, we introduce the selection mechanism, optimizer, and the balance of quality and diversity in Section 3.2 , Section 3.3, and Section 3.4, respectively. Finally, we review our algorithm and other QD algorithms in Section 3.5.

3.1 OVERVIEW

A general flow of EDO-CS is shown in Algorithm 1. It first initializes an archive A with queue properties and a maximum capacity of l, which contains the policies generated so far together with their corresponding cumulative rewards and behaviors, and a bandit B which is used to sample the weight of diversity during the reproduction process. Then it starts with randomly generating $4K$ policies and put them into the archive (line 1-4). In each iteration, EDO-CS uses K-means (Lloyd, 1982) to cluster the policies in the archive into K clusters based on their behaviors (line 8). Then, one specific policy is selected from each cluster (line 9). After getting K policies, EDO-CS samples the weights of diversity in the objective function from the bandit B, and uses ES optimizer to update the selected policies in parallel for T' iterations (line 11-15). The archive and bandit are updated after evaluating the offspring (line 16-17). EDO-CS repeats the above processes until the number of iterations reaches the total iterations T.

3.2 CLUSTERING-BASED SELECTION

Inspired by the successful application of clustering algorithms in problems such as subset selection (Denysiuk et al., 2014), we propose a clustering-based selection mechanism to select policies from the archive. Clustering is an unsupervised algorithm for finding natural groups in the feature space of input data. For a given data set $D = \{x_1, x_2, \cdots, x_n\}$, a clustering algorithm divides D into K clusters $C = \{C_1, C_2, \cdots, C_K\}$ where $\forall i \neq j, C_i \cap C_j = \emptyset$ and $D = \cup_{i=1}^{K} C_i$. K-means is one of the most popular clustering algorithms for its easy implementation and clear intuitionLloyd (1982). It aims to minimize the

Algorithm 1: EDO-CS

Input : number of selected policies K, total iterations T, updating iterations T', behavior characterization $b(\pi_\theta)$, archive size l, archive A, bandit B.
Output: archive A

```
// Warm up
for j = 1 : l do
    Randomly generate policy π_{θ_j};
    Get cumulative rewards R and behavior b(π_{θ_j}) by evaluating the policy π_{θ_j};
    Add (π_{θ_j}, R, b(π_{θ_j})) into archive A
end
t = 0;
while t < T do
    // Selection
    Use K-means to divide the policies in archive A into K clusters {C_k}_{k=1}^K;
    Select K policies {π_{θ_k}}_{k=1}^K, each one from a cluster;
    for k = 1 : K do
        // Update in parallel
        Sample λ_k from the bandit B;
        for i = 1 : T' do
            Set the objective function J(θ_k) = (1 − λ_k) E_{τ∼π_{θ_k}} [R(τ)] + λ_k Div(θ_k);
            Use ES to update θ_k (Equation 7)
        end
        Get cumulative rewards R and behavior b(π_{θ'_k}) by evaluating the updated policy π_{θ'_k};
        Update archive A and Bandit B
    end
    t = t + T'
end
```

within-cluster sum of squares (WCSS). Formally, the objective is to find:

$$\arg\min_{C} \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2, \tag{4}$$

where $\boldsymbol{\mu}_i$ is the mean of points in C_i.

In the selection process, we first use K-means to cluster the policies in the archive based on their behaviors, i.e., $\boldsymbol{x} = b(\pi_\theta)$ in Equation 4. Then, we select one specific policy from each cluster. More specifically, from the cluster where the best quality policy locates, we just select this policy. From the other cluster, we select uniformly from the M highest quality policies. Policies inside each cluster have similar behaviors, and each selected policy can be viewed as a representative of its cluster, which has the potential to reproduce the best quality policy with this type of behavior. This clustering-based selection mechanism effectively addresses the shortcomings of existing mechanisms. The selected policies have high quality and locate in different areas of the behavior space, so achieve both quality and diversity.

3.3 Evolution Strategies

In the implementation of our method, we use ES as the optimizer. Inspired by natural evolution, ES represents a broad class of population-based black-box optimization algorithms. The version we referred to

here was introduced in (Salimans et al., 2017), which was recently applied with large-scale deep learning architectures to modern RL problems. In ES, the population of parameter vectors θ is represented by a distribution $p_\phi(\theta)$, parameterized by ϕ. Given an objective function $J(\theta)$, ES seeks to optimize the expected objective value $\mathbb{E}_{\theta\sim p_\phi}[J(\theta)]$ using stochastic gradient ascent. Using the log-likelihood trick and estimating the expectation above over n samples, θ can be updated as follows:

$$\nabla_\phi \mathbb{E}_{\theta\sim p_\phi}[J(\theta)] \approx \frac{1}{n}\sum\nolimits_{i=1}^{n} J(\theta_i)\nabla_\phi \log p_\phi(\theta_i). \tag{5}$$

In ES, n samples are drawn with applying additive Gaussian noise to the current parameter vector, i.e., $\theta_i = \theta + \sigma\epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, I)$. With that notation, the above estimation of the gradient with respect to ϕ can be written as:

$$\nabla_\phi \mathbb{E}_{\theta\sim p_\phi}[J(\theta)] \approx \frac{1}{n\sigma}\sum\nolimits_{i=1}^{n} J(\theta + \sigma\epsilon_i)\epsilon_i. \tag{6}$$

The parameters will be updated with the following equation:

$$\theta_{t+1} = \theta_t + \frac{\eta}{n\sigma}\sum\nolimits_{i=1}^{n} J(\theta + \sigma\epsilon_i)\epsilon_i, \tag{7}$$

where η is the learning rate of the ES optimizer. In this paper, we set Equation 3 as the objective function of the policy in EDO-CS (line 13). However, it is still a problem to set the hyper-parameters λ so that the EDO-CS can be applied to different tasks. We will discuss this issue in the following subsection.

3.4 Adaptively balance of quality and diversity

Here, we model the problem of setting an appropriate λ in Equation 3 as a multi-armed bandit problem (Vermorel & Mohri, 2005). We treat each alternative λ as an arm of the bandit. Let $\Lambda = \{\lambda^1, \lambda^2, \cdots, \lambda^m\}$ denote the set of alternative arms.

Then, we use the Upper Confidence Bounds (UCB) algorithm (Auer, 2002) to select a specific arm in each iteration. In iteration t, the arm λ_t will be chosen according to the following equations:

$$\lambda_t = \arg\max_i \mu_t(\lambda^i) + c\sqrt{\frac{2\ln(t-1)}{N_t(\lambda^i)}},$$

where $\mu_t(\lambda_i)$ is the estimated mean rewards of arm λ_i, and $N_t(\lambda^i)$ is the number of times we selected arm λ_i. The estimated mean rewards reflect the current knowledge of the algorithm in a condensed form and guide further exploitation. The widths of the confidence bounds reflect the uncertainty of the algorithm's knowledge and will guide further exploration.

3.5 A review for QD algorithms

Here, we summarize our EDO-CS algorithm and other related works from the perspective of EAs, as shown in Table 1. We summarize these algorithms from four aspects: how the algorithm reproduces the offspring, how the algorithm selects the specific policy as the parents for reproduction, how many parents and offspring the algorithm will maintain in the population, and whether the algorithm selects the parents from the archive.

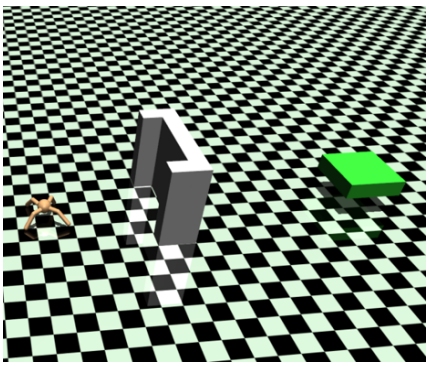
4 Experiment

To demonstrate the ability of EDO-CS to achieve both high quality and diversity, we conduct experiments on a variety of continuous control tasks from the *OpenAI Gym* library (Brockman et al., 2016), including

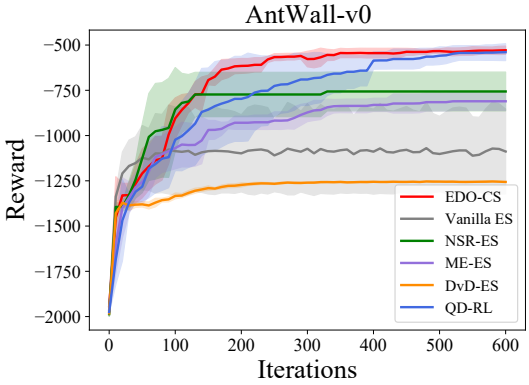
Table 1: A review for QD algorithms from the perspective of EAs.

Method	Reproduction	Selection	EAs type	From archive
Vanilla ES	Quality	All offspring	$(1+1)$	$\times$
NSR-ES	Quality and diversity	Probabilistic selection	$(K+1)$	$\times$
ME-ES	Quality or diversity	Biased selection	$(1+1)$	✓
DvD-ES	Quality and diversity	All offspring	$(K+K)$	$\times$
QD-RL	Quality or diversity	Pareto-based selection	$(K+K)$	✓
EDO-CS	Quality and diversity	Clustering-based selection	$(K+K)$	✓

deceptive, multi-modal, and standard MuJoCo environments. We empirically compare EDO-CS against Vanilla ES (Salimans et al., 2017), NSR-ES (Conti et al., 2018), DvD-ES (Parker-Holder et al., 2020), ME-ES (Colas et al., 2020), and QD-RL (Cideron et al., 2020). For fair comparisons, we use the ES optimizer with the same hyper-parameters in all considered algorithms. The population size K of these algorithms is set to 5 and remains the same in all tasks. For EDO-CS, the number of the candidates in each cluster $M = 2$, and the arms of the bandit B are $\{\lambda^1 = 0, \lambda^2 = 0.5\}$. Other settings of parameters can be found in Appendix A.1. To have a sample-wise fair comparison, the iterations in the x-axis are divided by the population size for algorithms that reproduce only one policy in each iteration (Vanilla ES, NSR-ES, ME-ES). We report the mean and standard deviations across six identical seeds (2016 - 2021) for all algorithms and all tasks.



(a) *AntWall-v0* environment



(b) Performance of different algorithms

Figure 2: The *AntWall-v0* environment and the performance of different algorithms in the *AntWall-v0* environment.

4.1 Deceptive Environment

First, we empirically compare these algorithms in an environment with deceptive rewards. We consider a modified environment based on *Ant-v2*, where a U-shape wall is in front of the ant, separating it from the target (green cuboid), as shown in Figure 2(a). The reward is a mixture of distance away from the target and costs for joint torques, encouraging the ant to walk to the target as fast as possible. Directly following the gradient of the expected cumulative rewards will lead the agent to walk directly forward and so get trapped by the wall. Here we use the final location (x, y) of the ant as the behavior characterization.

The results in Figure 2(b) show that both Vanilla ES and DvD-ES fail to get past the wall. These two algorithms adopt a straightforward selection mechanism, i.e., selecting all offspring, without any memory of

where it has already been explored. We can also observe that although NSR-ES converges fast, it finally gets trapped into a local optimum. It is expected because NSR-ES only reproduces one policy in each iteration updates the same policy for several iterations. It causes a rapid improvement in the quality of a particular policy but does not consider the whole behavior space, thus leading to a sub-optimal policy. Compared with the above algorithms, QD-RL can finally obtain the optimal policy at a slower speed, showing that the Pareto-based selection works better than the previous selection mechanisms but is still insufficient. EDO-CS achieves superior performance in terms of both convergence speed and the final result, disclosing the effectiveness of our clustering-based selection mechanism.

4.2 MULTI-MODAL ENVIRONMENTS

Next, we create two multi-modal environments to clearly verify that EDO-CS can efficiently find high-quality policies with diverse behaviors. The created *HalfCheetahFwdBwd* and *AntFwdBwd* environments are based on *HalfCheetah-v2* and *Ant-v2* respectively. In the *HalfCheetahFwdBwd* environment, instead of rewarding the agent for walking forward only, we assign rewards for both walking forward and backward. So there are two modals in this environment, i.e., walking forward and walking backward. We train policies in this multi-modal environment and test the policies in single-modal environments *HalfCheetahFwd* and *HalfCheetahBwd* respectively. We use the final location (x, y) of the agent as the behavior characterization. Similar settings in *AntFwdBwd* are used.

Table 2: Mean rewards obtained by different algorithms in multi-modal environments.

Environment	EDO-CS	QD-RL	ME-ES	DvD-ES	NSR-ES	Vanilla ES
HalfCheetahFwd	**4284**	2930	2700	-3419	1346	-5543
HalfCheetahBwd	**6548**	6013	5953	6353	5366	3911
AntFwd	**4617**	4291	4316	4507	1737	1911
AntBwd	**4697**	4164	4123	3498	3961	-851
Average Performance Ranking	**1**	2.75	3.25	3.5	4.75	5.75

To compare different algorithms more clearly, we organize the experimental results into a tabular form. Table 2 reports the mean of the cumulative rewards, together with the average performance rankings of all algorithms in four environments. We can observe that Vanilla ES can only learn a single modal, walking backward in *HalfCheetahFwdBwd* and walking forward in *AntFwdBwd*. This is because Vanilla ES maintains one policy, which can only take one action given a fixed state. It further demonstrates the importance of maintaining a population of policies. DvD-ES can learn both modals in *AntFwdBwd*, but a single modal in *HalfCheetahFwdBwd*. Although NSR-ES, ME-ES, and QD-RL can learn both modals in both environments, the quality of each modal is far from satisfying due to the inefficient selection mechanisms. Compared to other algorithms, EDO-CS can achieve higher quality in two modals of two environments, which shows that our method meets the goals of QD well.

4.3 ADDITIONAL RESULTS

In this section, we show the additional results of EDO-CS. More additional experimental results can be found in Appendix A.2.

Single-modal environments. To verify the effectiveness of EDO-CS in tasks that do not require much diversity, we conduct environments in standard MuJoCo environments. Here we adopt the action-based behavior characterization, which was also used in (Parker-Holder et al., 2020). The behavior characterization of a particular deterministic policy is represented in a vectorized form: $b(\pi_\theta) = \{\pi_\theta(\cdot \mid s)\}_{s \in \mathcal{S}}$, where $\mathcal{S}$ is the state space, and $\pi_\theta(\cdot \mid s)$ corresponds to the action taken in state s under policy π_θ. Since the state space

can be infinitely large, we randomly sample 20 states instead of all. We compare all algorithms used before, except ME-ES, which requires discretization of the behavior space and is hard to apply to high dimension behavior space. Figure 3 shows the results of different algorithms on four standard MuJoCo tasks. We observe that EDO-CS outperforms Vanilla ES in the final performance and achieves the fastest convergence compared to other population-based QD algorithms, i.e., DvD-ES and QD-RL.

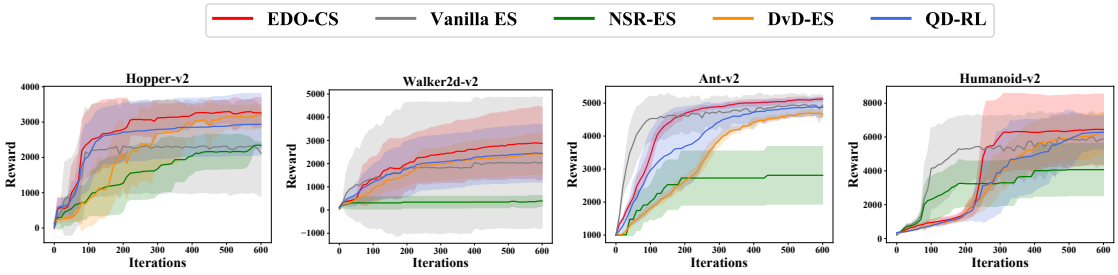


Figure 3: Performance of different algorithms in four standard MuJoCo environments.

Population size K. The population size K refers to the number of policies we select from the archive in each iteration. If the population size is too small, we may not be able to find all optimal policies with diverse behaviors; if the population size is too large, some policies with similar behavior will be selected and updated, harming the efficiency of the algorithm. How to set a proper population size K for different tasks is an interesting issue. Here we test the performance of EDO-CS under different population sizes, as shown in Table 3. For a fair comparison, we report the results under the same number of interactions with the environment rather than under the same iterations. We show that the default population size of $K = 5$ is appropriate in most environments.

Table 3: Mean rewards obtained by EDO-CS under different population sizes K in different environments.

Environment	$K = 3$	$K = 5$	$K = 7$	$K = 9$
AntWall-v0	-852	**-618**	-1082	-970
HalfCheetahFwdBwd	9101	**10274**	10221	10242
Walker2d-v2	1840	2453	3062	**3558**
Hopper-v2	2688	**3164**	2781	2856

5 Conclusion

QD algorithms aim to help RL to find a set of high-quality policies with diverse behaviors, solving challenging problems in complex environments. However, exciting algorithms can not fully guarantee both quality and diversity, mainly because of the inefficient selection mechanisms. To target this issue, we propose EDO-CS and demonstrate its ability to achieve both high quality and diversity on various tasks.

In this paper, we use ES as the optimizer. However, the proposed algorithm can also be equipped with the sample-efficient policy optimizer, which can further improve EDO-CS. The adaptive mechanism we used here is a simple one. Use state-of-the-art online-learning algorithms may better balance the importance of quality and diversity. In addition, how to decide the size of the population is also a matter of interest. One possible approach is to let the clustering algorithm determine the number of clusters by itself, and we leave these for future work.

REFERENCES

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

T. Bäck. *Evolutionary algorithms in theory and practice: Evolution Strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 1996.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *CoRR abs/1606.01540*, 2016.

K. Chatzilygeroudis, A. Cully, V. Vassiliades, and J.-B. Mouret. Quality-diversity optimization: A novel branch of stochastic optimization. 2020.

G. Cideron, T. Pierrot, N. Perrin, K. Beguir, and O. Sigaud. QD-RL: Efficient mixing of quality and diversity in reinforcement learning. *CoRR abs/2006.08505*, 2020.

C. Colas, V. Madhavan, J. Huizinga, and J. Clune. Scaling map-elites to deep neuroevolution. In *Proceedings of the 22nd Genetic and Evolutionary Computation Conference (GECCO)*, pp. 67–75, Cancún, Mexico, 2020.

E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. 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 32 (NeurIPS)*, pp. 5032–5043, Montréal, Canada, 2018.

A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553): 503–507, 2015.

K. Deb. Multi-objective optimisation using evolutionary algorithms: An introduction. In *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pp. 3–34. Springer, 2011.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

R. Denysiuk, L. A. Costa, and I. A. E.-Santo. Clustering-based selection for evolutionary many-objective optimization. In *Proceedings of the 13th Parallel Problem Solving from Nature (PPSN)*, pp. 538–547, Ljubljana, Slovenia, 2014.

S. Kumar, A. Kumar, S. Levine, and C. Finn. One solution is not all you need: Few-shot extrapolation via structured MaxEnt RL. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 8198–8210, Vancouver, Canada, 2020.

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)*, pp. 211–218, Dublin, Ireland, 2011a.

J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011b.

Y. Li. Deep reinforcement learning: An overview. *CoRR abs/1701.07274*, 2017.

F.-Y. Liu and C. Qian. Prediction guided meta-learning for multi-objective reinforcement learning. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pp. 2171–2178, Kraków, Poland, 2021.

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. Learning to navigate in complex environments. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts. Effective diversity in population based reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pp. 18050–18062, Vancouver, Canada, 2020.

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

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR abs/1703.03864*, 2017.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pp. 437–448, Porto, Portugal, 2005.

Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, F.-F. Li, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3357–3364, Singapore, 2017.

A APPENDIX

A.1 HYPER-PARAMETERS

This section will introduce the hyper-parameters used in our experiments. For fair comparisons, all the hyper-parameters are set to the same among the different algorithms in each environment.

Table 4: Hyper-parameters for deceptive and multi-modal environments.

Environment	*AntWall-v0*	*HalfCheetahFwdBwd*	*AntFwdBwd*
Population size M	5	5	5
Archive size l	20	100	100
Update iterations T'	10	5	10
σ in ES	0.025	0.03	0.025
η in ES	0.015	0.02	0.015

Table 5: Hyper-parameters for single-modal environments

Environment	*Hopper-v2*	*Walker2d-v2*	*Ant-v2*	*Humanoid-v2*
Population size M	5	5	5	5
Archive size l	30	30	30	20
Update iterations T'	5	5	10	10
σ in ES	0.025	0.025	0.025	0.0075
η in ES	0.02	0.03	0.015	0.02

A.2 ADDITIONAL RESULTS

Clustering algorithm. An essential step in EDO-CS is to cluster policies in the behavior space. In this paper, we adopt the K-means clustering algorithm because of its wide use and clear intuition. However, there is still a question of whether the choice of this has an impact on performance. As such, we conducted experiments over various clustering algorithms, including Hierarchical Agglomerative Clustering (HAC), K-means, Balanced Iterative Reducing, and Clustering Using Hierarchies (BIRCH), and Spectral Clustering. Figure 4 shows that in most cases, these clustering algorithms have good performance, except BIRCH. It is because that BIRCH is suitable for a large number of samples, while the number of policies in the archive is relatively small. Thus we cannot get good clustering results, which affects the final performance of the algorithm.

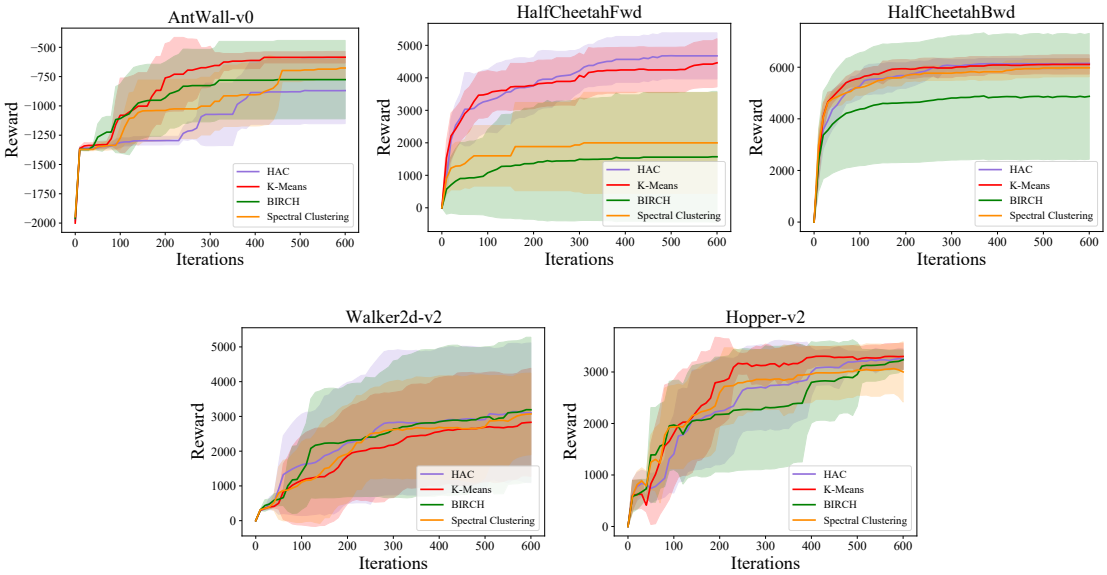


Figure 4: Performance of EDO-CS equipped with different clustering algorithms

Update iterations T'. We also study the effect of update iterations, which means the number of times we update the policy after selecting it from the archive. We have tried to update the selected policy for only one iteration and then add it into the archive, but it fails. This indicates that the update iteration should not be set too small. Otherwise, the difference between solutions will be insignificant and thus cause ineffective selection. We conduct experiments over several appropriate values. Table 6 shows the results.

Table 6: Performance of EDO-CS under different update iterations T'

Environment	$T' = 5$	$T' = 10$	$T' = 15$	$T' = 20$
HalfCheetahFwd	4462	4216	4911	**4926**
HalfCheetahBwd	6407	**6527**	6285	6203
Walker2d-v2	2830	**4870**	4362	4676
Hopper-v2	3301	3566	**3578**	3559

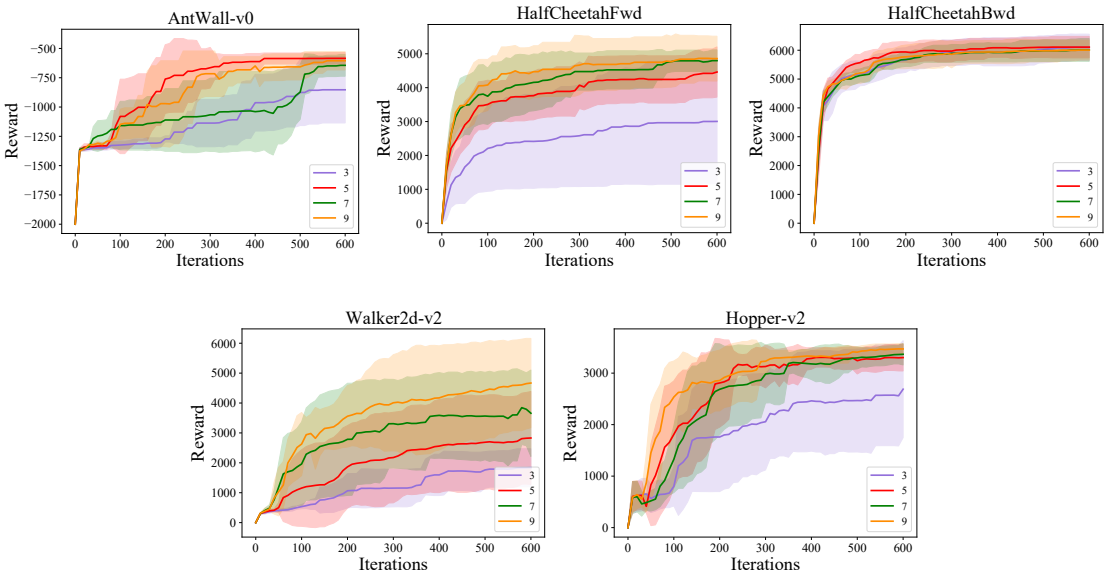


Figure 5: Performance of EDO-CS under different population sizes K.

Population size K. Figure 5 shows the performance of EDO-CS under four different population sizes with the same iterations. Contrast to the results in Table 3, the different settings in Figure 5 have different time-steps, i.e., the interaction times with environments. Larger population sizes have a greater number of interactions with the environment, resulting in generally better performance

Archive size l. The archive is also an important part of our algorithm. On the one hand, we have to select policies from it in each iteration. On the other hand, when updating policies, the diversity of a specific policy should be calculated according to the policies stored in the archive. If the archive size is set too large, it will contain lots of low-quality policies encountered previously. Selecting these policies will do harm to the efficiency. If the archive size is set too small, the useful information may be forgotten, affecting the performance. Figure 6 shows the results under different archive sizes. How to adaptively set the archive size is still an interesting issue in the fields of QD.

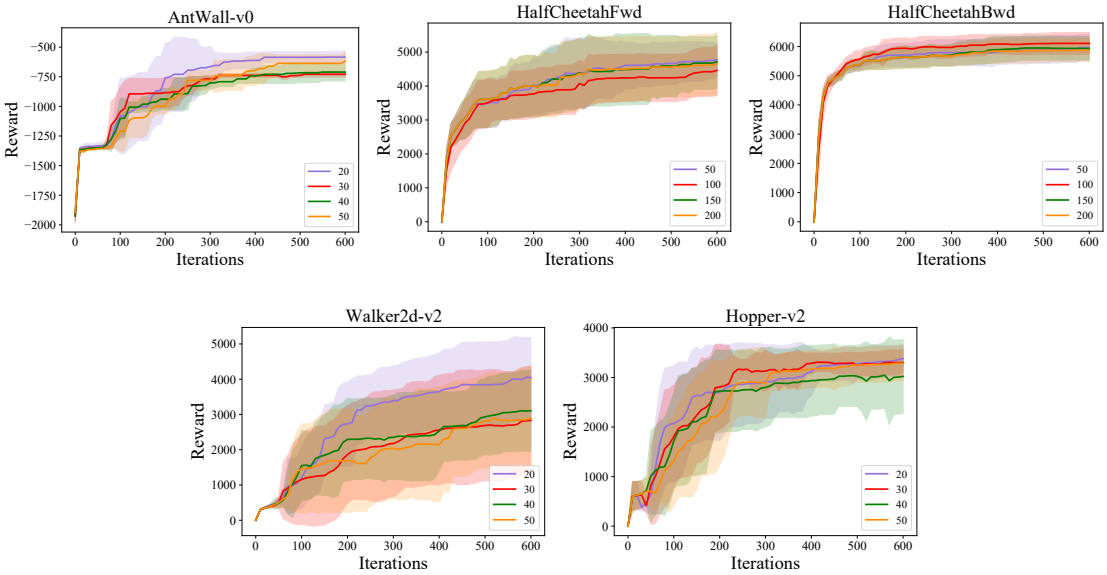


Figure 6: Archive size