Neurocomputing 556 (2023) 126628

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# A survey on Evolutionary Reinforcement Learning algorithms

Qingling Zhu, Xiaoqiang Wu, Qiuzhen Lin, Lijia Ma, Jianqiang Li, Zhong Ming, Jianyong Chen *

*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China*

ARTICLE INFO

*Keywords:*
Evolutionary algorithm
Reinforcement learning
Evolutionary reinforcement learning
Policy optimization

ABSTRACT

Reinforcement Learning (RL) has proven to be highly effective in various real-world applications. However, in certain scenarios, Evolutionary Algorithms (EAs) have been utilized as an alternative to RL algorithms. Recently, Evolutionary Reinforcement Learning algorithms (ERLs) have emerged as a promising solution that combines the advantages of both RL and EA. This paper presents a comprehensive survey that encompasses a majority of the studies in this exciting research area. We classify these ERLs according to the EA used in their frameworks and analyze the strengths and limitations of various EA components and combination schemes. Additionally, we conduct several experiments to evaluate the performance of some representative ERLs. By categorizing the different approaches and assessing their effectiveness, the paper can assist researchers and practitioners in selecting the most suitable method for their particular application.

## 1. Introduction

Reinforcement learning (RL) is a prominent branch of machine learning that has achieved significant success in training agents to perform various tasks at or above human-level proficiency. For example, the Deep Q-network (DQN) has demonstrated comparable performance to professional human game testers in Atari [1]. Additionally, AlphaGo [2] and OpenAI Five [3] have triumphed over world champions in the game of Go and Dota2, respectively. The incorporation of reinforcement learning with deep neural networks, commonly referred to as Deep Reinforcement Learning (DRL), is a primary factor contributing to RL's accomplishment. Despite the successful application of DRL in various domains, it is not yet a common practice to use DRL to solve real-world problems. One significant reason for this is the lack of effective exploration in DRL, which often leads to premature convergence to local optima [4].

Evolutionary Reinforcement Learning algorithms (ERLs) offer a powerful approach that effectively merges the benefits of both RL algorithms and Evolutionary Algorithms (EAs), thus minimizing their individual drawbacks. In most cases, ERLs typically take up a key component of RL algorithms, such as an agent, action, or parameter, and treat it as an individual within a population. By subjecting these individuals to iterative updates using EAs, they evolve and diversify, leading to the emergence of highly performing individuals that enhance the overall performance of the RL algorithms. Consequently, ERLs provide a robust framework for enhancing the efficiency and effectiveness of RL algorithms.

This paper provides a comprehensive survey of ERLs, which are classified according to the EAs used in their frameworks. The classification is based on the fact that the primary difference between these algorithms lies in the design of their EA components, as the RL algorithms used in ERLs are typically similar or identical. The EAs commonly employed in ERLs include Genetic Algorithm (GA) [5], Evolution Strategy (ES) [6], Particle Swarm Optimization (PSO) [7], and Genetic Programming (GP) [8]. Given the prevalence of GA and ES in ERLs research, we categorize them as ERLs with GAs and ERLs with ESs, respectively. For other ERLs that do not fit into these categories, including a few that do not specify the EAs they use, we group them as ERLs with other EAs. We provide a detailed discussion and analysis of the advantages and limitations of each type of ERLs. Moreover, we conduct several experimental studies to provide empirical evidence for the effectiveness of various ERLs.

The organization of this paper is as follows. Section 2 provides an overview of both EAs and RL algorithms. In Section 3, we present a comprehensive review of ERLs that utilize various EAs, categorized according to the EA used in their frameworks. Section 4 presents an empirical analysis of the performance of different ERLs on six continuous control tasks and four classic control problems. Finally, in Section 5, we summarize our findings and suggest some future directions.

## 2. Background

### 2.1. Evolutionary algorithm

EAs are population-based heuristic algorithms that are inspired by natural evolution [9]. The overall framework of EAs consists of three

* Corresponding author.
*E-mail address:* jychen@szu.edu.cn (J. Chen).

https://doi.org/10.1016/j.neucom.2023.126628
Received 21 April 2023; Received in revised form 25 June 2023; Accepted 24 July 2023
Available online 5 August 2023

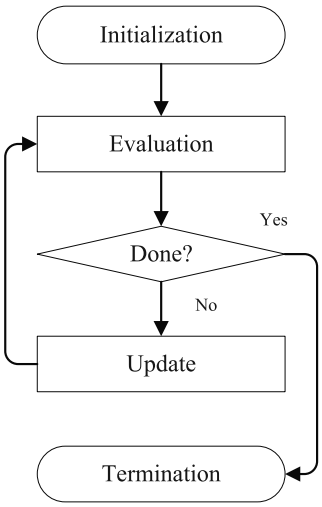



**Fig. 1.** The overall framework of EAs.

key procedures: initialization, evaluation, and update, as illustrated in Fig. 1. During initialization, the population and parameters are set up. In the evaluation stage, individuals within the population are assessed to determine their fitness values. Based on the evaluation results, individuals in the population are updated using a specific method that depends on the chosen EA. These evaluation and update procedures are iteratively executed until the termination criterion is met. In the following sections, we provide a brief overview of the most commonly used EAs in ERLs.

#### 2.1.1. Genetic algorithm

GAs are a type of heuristic search algorithm that mimics natural selection and biological evolution processes. The update procedure in GA typically involves three genetic operators: selection, crossover, and mutation. The primary objective of the selection operator is to choose the parents of the next generation [10]. This selection process primarily relies on the fitness values of individuals and usually includes some random operations to maintain the diversity of the population. Common selection methods include tournament selection [11] and roulette wheel selection [12], among others. After the selection process, parents are combined using the crossover operator to produce the offspring for the next generation. For instance, in single-point crossover [13], the chromosomes of parents are randomly divided into two parts, and one of which is swapped. Finally, individuals in the population are subject to mutation with a specific probability. In many GAs, individuals with the highest fitness values are protected from mutation, which is referred to as elitism [14]. In ERLs, mutation operators are typically designed based on Gaussian mutation [13].

#### 2.1.2. Evolution strategy

ES is another type of EA that employs a unique update procedure. ES utilizes the normal distribution calculated from the elite individuals in the population to generate the next generation. The Cross-Entropy Method (CEM) is a well-known method in this field [15] and is widely used in ERLs. For CEM, the mean and covariance matrix of the distribution are calculated from the top $K_e$ individuals with the highest fitness values in the population, as follows:

$$\mu_{new} = \sum_{i=1}^{K_e} \lambda_i z_i, \tag{1}$$

$$\Sigma_{new} = \sum_{i=1}^{K_e} \lambda_i (z_i - \mu_{old})(z_i - \mu_{old})^T + \epsilon \mathcal{I}, \tag{2}$$

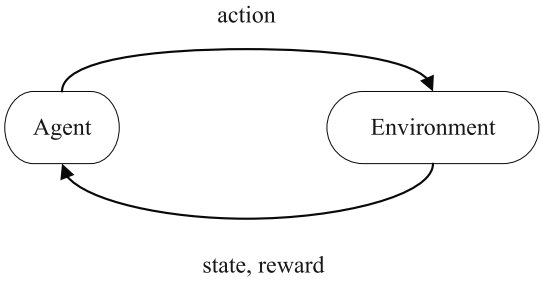



**Fig. 2.** The interaction between the agent and environment.

where $z_i$ indicates the individual $i$, $\lambda_i$ is the weight for individual $i$. The value of $\lambda_i$ is generally set to $\lambda_i = \frac{1}{K_e}$ or $\lambda_i = \frac{log(1+K_e)/i}{\sum_{i=1}^{K_e} log(1+K_e)/i}$ [16]. In the former setting, the weights of all individuals are identical, while in the latter setting, the weights of individuals with high rank are higher. In CEM, all new individuals are sampled from the distribution $\mathcal{N}(\mu, \Sigma)$.

In practical implementations, the computational complexity of the covariance matrix $\Sigma$ is high, so it is often assumed to be diagonal [17]. Additionally, Eq. (2) is replaced by the following calculation:

$$\Sigma_{new} = \sum_{i=1}^{K_e} \lambda_i (z_i - \mu_{old})^2 + \epsilon \mathcal{I}, \tag{3}$$

### 2.2. Reinforcement learning

Reinforcement learning can be modeled as a Markov Decision Process (MDP) [18]. At each time step $t$, the agent receives a state $s_t$ from the environment, and selects an action $a_t$ to perform. Next, the environment provides the agent with the next state $s_{t+1}$ and a reward $r_t$, as illustrated in Fig. 2. This process of interaction is repeated until the agent reaches a termination state.

The cumulative discounted reward can be expressed mathematically as $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$, where $\gamma$ is the discount factor and $T$ is the length of one episode. The primary goal of RL algorithms is to discover the optimal policy that maximizes the cumulative discounted reward $R_t$. The remainder of this section provides a brief introduction to the RL algorithms used in ERLs.

#### 2.2.1. Deep deterministic policy gradient

The Deep Deterministic Policy Gradient (DDPG) is an off-policy, model-free RL algorithm that extends the idea of DQN to continuous action spaces [19]. The actor-critic architecture of DDPG involves an actor network $\mu$ that selects actions and a critic network $Q$ that approximates the action-value function. Similar to DQN, DDPG employs target networks $\mu'$ and $Q'$ to stabilize training. During the interaction, the agent stores the tuple $(s_t, a_t, r_t, s_{t+1})$ in an experience replay buffer $R$. Later, a batch of tuples is randomly sampled from $R$ to update the policy and critic networks. The critic network is updated by minimizing the loss function below.

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2, \tag{4}$$

where $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$, $\theta$ represents the parameters of network and $N$ represents the batch size. Then the actor is updated by the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}. \tag{5}$$

Additionally, the soft target update technique is employed to enhance learning stability.

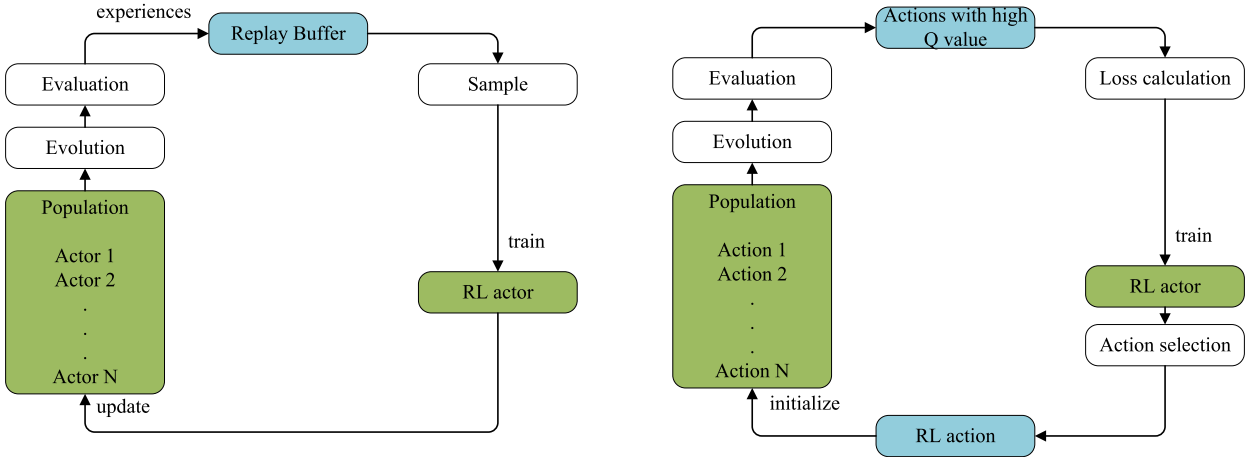



**Fig. 3.** Two popular frameworks in ERLs. Left: population of agents; Right: population of actions.

*2.2.2. Twin delayed deep deterministic policy gradient*

DDPG overestimates the value of action when neural network is used to approximate the value function [20]. To address this approximation error, Twin Delayed Deep Deterministic Policy Gradient (TD3) [21] maintains two separate value function networks and takes the minimum between the two estimates. Consequently, $y_i$ in the critic loss is computed as follows.

$$y_i = r_i + \gamma min_{j=1,2} Q'_j(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'_j}). \quad (6)$$

In order to further reduce the cumulative estimation error, TD3 reduces the update frequency of the policy and target networks, which is referred to as the delaying policy update strategy. Furthermore, TD3 employs a target policy smoothing strategy to smooth the value estimation of the target network. The core concept behind this strategy involves introducing noise to the next action during the estimation of its value. This approach effectively smooths value estimation among similar actions, thereby improving the overall performance of the learning algorithm.

## 3. A survey of evolutionary reinforcement learning algorithms

In this section, a comprehensive introduction to ERLs is provided. Different ERL algorithms are presented based on their used EA. Section 3.1 covers ERLs that utilize GA, Section 3.2 describes ERLs that employ ES, and Section 3.3 presents ERLs that use other EAs.

### 3.1. ERLs with GA

GAs have shown promise as an alternative for optimizing neural networks in reinforcement learning [22]. To further exploit the benefits of GAs, several approaches that integrate GAs with RL have been proposed and demonstrated superior performance compared to original RL algorithms or GAs alone. In this section, we introduce these algorithms based on the type of individuals within the population.

*3.1.1. Population of agents*

Combining EAs and RL by using agents as individuals is a direct approach that enables EAs to optimize and generate policies. In this approach, ERLs typically use a population of agents optimized by EA and an additional agent optimized by RL, as illustrated in Fig. 3. In this framework, a population of agents is used to provide diverse experiences for the training of RL agents. In addition, the RL agent updated by the gradient method can be utilized to update the population. The fitness value is determined by the sum of rewards obtained in a single episode, and evaluating an agent involves allowing it to interact with the environment. During the evaluation process of individuals in the population, many experiences or tuples can be stored in the replay buffer. However, if the RL algorithm used in ERLs is on-policy, these experiences may not be fully exploited by the RL agent, as they could violate the on-policy property. Therefore, off-policy RL algorithms, particularly DDPG and its variants [21], which exhibit excellent performance in continuous action domains, are more popular in ERLs than on-policy RL algorithms.

Evolutionary Reinforcement Learning (ERL) [4] is one of the most popular algorithms in this kind of combination. The RL algorithm used in ERL is DDPG and its actor-network is selected as an individual in EAs. The population of actors can provide diverse experiences for the RL actor's training, and the updated RL actor will be inserted into the population as a candidate for the next generation. In this combination scheme, GA and DDPG complement each other, achieving better performance than the original GA and RL algorithms. The GA in ERL consists of tournament selection, crossover, and mutation operators. The process of producing the next population is as follows. First, the population is evaluated, and elitists with the set of highest fitness values are selected. Then, tournament selection is applied to the population, except for the elitists. After that, the elitists and selected individuals become parents and produce new individuals by crossover. Specifically, the new individuals are obtained by exchanging the partial weights of the parents. Finally, all individuals except for elitists are mutated by mutation, that is, adding Gaussian noise to their weights. Both crossover and mutation operators are executed with a certain probability. Throughout the process of updating the population, elitism protects the bunch of best individuals from damage, while the genetic operators maintain the diversity of the population.

Building on the GA in ERL [4], several variants have been proposed to address different challenges in RL. Collaborative Evolutionary Reinforcement Learning (CERL) [23] employs a portfolio of policies that explore diverse regions of the solution space together to mitigate the sensitivity of hyperparameters in RL algorithms. Additionally, the computation resources are dynamically allocated to enhance algorithm efficiency. CERL utilizes the TD3 [21] algorithm, and it outperforms TD3 with different discount factors. Besides, ERL-Re$^2$ [24] presents a two-scale representation that includes share nonlinear state representation and independent linear policy representation. In ERL-Re$^2$, the genetic operators are only applied to the linear policy representation of individuals, which reduces the search space of GA. Based the two-scale representation, ERL-Re$^2$ utilizes a value function approximator to estimate the fitness values and improve the sample efficiency. Moreover, Multiagent Evolutionary Reinforcement Learning (MERL) [25] extends ERL [4] to multiagent reinforcement learning. Since sparse rewards are not conducive to the learning of RL algorithms, MERL uses EA and TD3 to optimize the sparse team rewards and dense agent-specific

rewards, respectively. Similar to ERL, RL agents are periodically added to the population in MERL to transfer gradient-based optimization information to EA. Experimental results on a range of coordination benchmarks show that MERL outperforms MADDPG [26]. Multiagent Evolution via Dynamic Skill Selection (MAEDyS) [27] employs a similar algorithm framework as MERL, but the policies in the population only learn to optimize the selection of local skills. The selected RL policy then executes the specific action. MAEDyS has been evaluated in complex multiagent coordination environments and has shown superior performance compared to previous methods. Subsequently, Representation Asymmetry and Collaboration Evolution (RACE) [28] extends the idea of ERL-Re$^2$ to MARL. RACE maintains a population of teams and agents in teams own independent policy representation and shared observation representation. In addition, RACE proposes Value-Aware Mutual Information Maximization to make the shared observation representation learn more information from the global states with high values. Experimental results show that RACE outperforms MERL and other baselines on a range of continuous and discrete tasks.

Although ERL [4] has demonstrated that combining GA and RL methods can outperform original algorithms, the genetic operators used in ERL are destructive, which limit the performance of ERL. To address this problem, Proximal Distilled Evolutionary Reinforcement Learning (PDERL) is proposed [29]. In PDERL, two novel genetic operators are designed based on backpropagation: proximal mutation and distillation crossover. The experiences of individuals are stored in their own replay buffer, called genetic memory. The proximal mutation operator is based on the idea of safe mutation [30], which uses the gradient of actions over a batch of transitions from the genetic memory to compute the sensitivity of weights. Then, the sensitivity is used to scale the mutation strength. Specifically, the more sensitive the weight is, the smaller the mutation strength becomes. Therefore, the behavior of child policies is proximal to their parent's behavior. The distillation crossover operator copies one of the parents as a child and then utilizes the genetic memory of the parents to train the child in a way similar to Imitation Learning [31]. The two genetic operators are integrated with the framework of ERL to form PDERL. Experiments have showed that PDERL outperforms ERL, TD3, and PPO [32] in several Mujoco test environments [33,34]. To improve the sample efficiency of ERL and PDERL, the Surrogate-assisted Controller (SC) [35] is proposed to partially replace the expensive policy evaluation. SC is integrated with ERL and PDERL to form two new frameworks named SERL and SPDERL, which perform better than the original algorithms. In addition, some algorithms try to avoid unnecessary evaluations of the population, which only evaluate the population when RL agent learns slowly [36].

Moreover, some hybrid approaches attempt to combine different methods to take advantages of their strengths. One such approach is Competitive and Cooperative Heterogeneous Deep Reinforcement Learning (C2HRL) [37], which uses a pool of heterogeneous agents to induce competition for computation resources and promote exploration in diverse regions of the solution space. C2HRL employs TD3 and Soft Actor-Critic (SAC) [38] as its RL algorithms, and uses random perturbations as the mutation operations for the neural network parameters. The empirical results on continuous control problems demonstrate that C2HRL outperforms TD3, SAC, and CERL. Soft Updates for Policy Evolution (Supe-RL) [39] is a hybrid method that combines GA and Rainbow [40], which is a value-based RL algorithm. Supe-RL proposes genetic soft update where the RL agent is updated by the best individual of the population in a way similar to soft update [41]. Supe-RL does not use a crossover operator, and the mutation operator is similar to that of ERL. However, all new individuals in the next generation are mutated from the RL agent. Genetic-Gated Network (G2N) [42] uses binary genetic genes to control the activation of neurons in the hidden layer, which is inspired by the dropout method [43] in machine learning. G2N is integrated into two RL frameworks (Synchronous Advantage Actor-Critic (A2C) [44] and PPO) to form G2AC and G2PPO. Experimental results demonstrate that all these hybrid approaches achieve better performance than the original algorithms.

The algorithms discussed above do not explicitly maintain population diversity, but some hybrid algorithms do use diversity maintenance techniques, such as Novelty Search (NS) [45] and Quality Diversity (QD) [46]. NS-RL [47] uses NS to improve the diversity of agents and combines GA and DDPG. It calculates the novelty of agents and replaces less novel agents with new agents, while protecting elitists and optimizing all selected individuals by DDPG. In NS-RL, all individuals except for elitists undergo mutation, and the crossover operation is omitted. Policy Gradient Assisted MAP-Elites (PGA-MAP-Elites) [48] use a QD algorithm called MAp-Elites [49] to maintain population diversity. In PGA-MAP-Elites, half of the population undergoes mutation through policy gradient, while the other half is mutated in a way similar to Differential Evolution (DE) [50]. After each iteration, the individuals in the population are used to update the archive in MAP-Elites. Furthermore, Quality-Diversity Policy Gradient (QD-PG) [51] also employs MAP-Elites to ensure population diversity. In QD-PG, half of the population undergoes optimization using TD3 to enhance quality, while the other half is updated using TD3 with novelty rewards to promote diversity. This approach replaces the genetic mutation operator.

#### 3.1.2. Population of others

Sample-Efficient Automated Reinforcement Learning (SEARL) [52] maintains a population of agents and corresponding hyperparameters. During each iteration, all individuals are evaluated and selected through tournament selection. The selected individuals are then mutated by multiple mutation operators, which randomly modify the network weights, activation function, size of neural network, and hyperparameters. These operators are chosen with equal probability. Notably, SEARL optimizes both the hyperparameters and neural architecture simultaneously, thereby enhancing the efficiency of hyperparameter optimization in Population-Based Training (PBT) [53].

### 3.2. ERLs with ES

Evolution strategies have been shown to be a scalable alternative to RL algorithms [54], and are widely recognized as one kind of efficient EA [17]. To fully leverage the advantages of both ES and RL algorithms, several novel combinations have been proposed and have demonstrated competitive performance over ERLs with GA. This section introduces these algorithms based on the type of individuals in the population.

#### 3.2.1. Population of agents

Cross-Entropy Method Reinforcement Learning (CEM-RL) [17] uses agents as individuals in the population, but instead of inserting policy gradient optimized agents into the population, CEM-RL applies a gradient update to half of the population. In each iteration, all individuals are produced from the distribution first. Then, half of the population is updated by policy gradient. Finally, all individuals are evaluated and the distribution is updated with the better half of the population. In CEM-RL, CEM is combined with DDPG and TD3, called CEM-DDPG and CEM-TD3. Both combinations perform better than the original algorithms and ERL in several robot locomotion simulation environments. Using a similar framework to CEM-RL, CEM-ACER [55] replaces the RL component of CEM-RL with Actor-Critic with Experience Replay (ACER) [56] and outperforms its components in a range of benchmarks. In addition, Asynchronous Evolution Strategy-Reinforcement Learning (AES-RL) [57] uses an asynchronous scheme to combine ES and TD3 for parallel computing. AES-RL exploits the benefits of the parallelism in ES and reduces the wall clock time of training on the same hardware configuration. Moreover, it also shows better performance than CEM-RL [57]. Based on the framework of CEM-RL, Diversity Evolutionary Policy Deep Reinforcement Learning (DEPRL) [58] uses the maximum mean difference (MMD) to measure the distance between different

policies and improve the diversity of population. The experimental results show that DEPRL has achieved improved performance compared to CEM-TD3 in the Mujoco test environment.

Evolution-based Soft Actor-Critic (ESAC) [59] adopts a similar combination mechanism to ERL, but integrates SAC and ES into the ERL framework. Additionally, it utilizes soft winner selection and genetic crossover in hindsight, which distinguishes it from other hybrid algorithms with ES. The crossover operation replaces weight vectors randomly. Moreover, ESAC proposes Automatic Mutation Tuning (AMT) to reduce the sensitivity of algorithm to hyperparameters. The experimental results show that ESAC outperforms other compared RL algorithms in several challenging locomotion tasks.

Cooperative Heterogeneous Deep Reinforcement Learning (CHDRL) [60] is a hierarchical framework that combines the advantages of different RL algorithms. This framework consists of two levels of agents. The upper-level agents are global off-policy agents that can utilize experiences from all agents, while the low-level agents are local on-policy or evolutionary agents that only explore the local area. The global agents guide the learning of local agents, allowing them to benefit from the high sample efficiency of the global agents. In addition, the local agents can effectively explore their local areas and their experiences enable the global agents to benefit from their efficiency and stability. In CHDRL, the authors propose a practical algorithm called CPSC that combines Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Cross-Entropy Method (CEM). The experimental results show that CPSC outperforms its individual components in various continuous control benchmarks.

#### 3.2.2. Population of actions

In addition to using agents as individuals, there are also algorithms that select actions as individuals. Unlike prior methods that optimize the policies directly, these algorithms utilize EAs to search for superior actions to promote the learning of RL algorithms, as depicted in Fig. 3. In this framework, EAs are utilized to search actions with high Q values to guide the learning of RL agent. Besides, the action selected by RL agent can be used to initialize the population. Therefore, these algorithms have no need to consider whether the randomness of EAs might destroy the functionality of neural networks. However, the difficulty of searching for superior actions increases as the dimension of action space increases.

Cross-Entropy Guided Policies (CGP) [61] uses the CEM algorithm to sample the action with the maximum value to guide the learning of policy gradient methods. Additionally, a deterministic policy is trained by imitating the behavior of the CEM policy. Experimental results demonstrate that CGP achieves better performance than other RL algorithms. Soft Actor-Critic with Cross-Entropy Policy Optimization (SAC-CEPO) [62] applies a similar idea to the stochastic policy RL algorithm (SAC). To reduce the computational burden, SAC-CEPO decouples the policy network of SAC into two networks. One network outputs the mean of the Gaussian distribution, and the other outputs the variance. Self-guided and self-regularized actor-critic (GRAC) [63] uses the actor network to output an initial Gaussian distribution before searching by CEM. Then, CEM is executed to search for an action with a higher Q value based on the results of the actor network. In this way, CEM can perform well in some problems with high-dimensional action space.

### 3.3. ERLs with other EAs

There are several ERLs that make use of other evolutionary algorithms (EAs), such as PSO and GP. PSO is a swarm intelligence optimization algorithm, that was developed by Kennedy and Eberhart [7,64]. In this algorithm, all individuals in the population are updated by taking guidance from both the historically best individual and the globally best individual. GP is an optimization algorithm that evolves individuals represented as trees [8]. Similar to GA, GP also employs selection, crossover, and mutation operations to generate new individuals. Moreover, some hybrid algorithms do not explicitly specify the type of EA used, but they are all population-based. This section will also introduce these methods.

Evolutionary Action Selection-Twin Delayed Deep Deterministic Policy Gradient (EAS-TD3) [65] combines PSO and TD3, in which PSO is used to optimize the action chosen by the policy network. At each timestep, an evolutionary action is obtained by PSO and the search of PSO is based on the action from TD3 policy. Besides, the policy network is updated to imitate the evolutionary action after the update of policy gradient. Using the evolutionary action to guide the learning of RL algorithms, EAS-TD3 shows superior performance over the original TD3.

Evolutionary-Driven Reinforcement Learning (Evo-RL) [66] utilizes Behavior Trees (BT) [67] and neural networks to represent policies. In Evo-RL, the BT is evolved using GP, while the neural networks are optimized by RL algorithms. The fitness values of individuals are evaluated based on the overall performance of the complete policy, since BT only represents a partial policy and is difficult to evaluate in isolation. Evo-RL combines BT with DQN and PPO, resulting in eDQN and ePPO. Experimental results show that both algorithms outperform the original RL algorithm in three classic control problems.

Population-Based Training (PBT) [53] employs a population of agents to optimize the hyperparameters of algorithms. Similar to GA, PBT uses a selection operator to exploit the information of excellent individuals in the population. It also utilizes random mutation to produce diverse hyperparameters for exploration. Through hyperparameter optimization, PBT can achieve better performance than baseline algorithms trained for the same number of steps.

Multi-Path Policy Optimization (MPPO) [68] maintains a population of diverse policies to improve exploration in on-policy RL algorithms. At each iteration of MPPO, a candidate policy from the population is selected based on performance and entropy. This selection rule strikes a balance between exploration and exploitation. The selected policy is then evaluated, and transitions are obtained during interactions with the environment. The policy is updated using these transitions while still adhering to the on-policy property. Finally, the selected policy replaces itself in the population to update the population. MPPO is implemented with both PPO and TRPO, and both combinations outperform other compared exploration methods. Furthermore, a theoretical guarantee of stable performance for Multi-Path TRPO is presented in MPPO.

Population-guided Parallel Policy Search-Twin Delayed Deep Deterministic Policy Gradient (P3S-TD3) [69] proposes a population-guided parallel scheme to enhance the performance of off-policy RL algorithms. In P3S-TD3, each individual has its own policy network and value function network, while many individuals in other ERLs only include a policy network. All individuals are updated by TD3 in each iteration. Additionally, an augmented loss function is designed to guide the learning of individuals, and this loss is influenced by the previous best policy.

## 4. Experimental studies on ERLs

### 4.1. Compared algorithms

In this section, we compare and analyze the performance of several existing ERLs on continuous control problems. Specifically, we evaluate four popular ERLs with different combination schemes, along with two state-of-the-art RL algorithms. Below, we provide a brief introduction to each of these algorithms:

(1) ERL [4]: ERL is one of the most popular hybrid algorithms that combines policy gradient and GAs and it is the first algorithm as the combination of EA and RL to achieve excellent performance on robot locomotion tasks.

**Table 1**
Basic information of Mujoco test environments and classic control environments.

| | Environment | State dimension | Action dimension | Goal |
|---|---|---|---|---|
| Mujoco | HalfCheetah | 17 | 6 | Make the biped cheetah run forward as fast as possible |
| | Swimmer | 8 | 2 | Make the robot move forward as fast as possible |
| | Reacher | 11 | 2 | Make the robot move close to a target that is spawned at a random position |
| | Hopper | 11 | 3 | Make the one-legged robot move forward as fast as possible |
| | Ant | 111 | 8 | Make the four-legged robot move forward as fast as possible |
| | Walker2d | 17 | 6 | Make the two-legged figure move forward as fast as possible |
| Classical | BipedalWalker | 24 | 4 | Make the biped robot move forward |
| | LunarLander | 8 | 2 | Make the spaceship safely land between two flags |
| | MountainCar | 2 | 1 | Make the car reach the top of the right hill |
| | Pendulum | 3 | 1 | Make the pendulum reach an upright position |

**Table 2**
The hyperparameters for all compared algorithms.

| Hyperparameters | ERL | CERL | CEM-RL | PDERL | DDPG | TD3 |
|---|---|---|---|---|---|---|
| Discount factor | 0.99 | * | 0.99 | 0.99 | 0.99 | 0.99 |
| Actor learning rate | 5e−5 | 1e−3 | 1e−3 | 5e−5 | 5e−5 | 3e−4 |
| Critic learning rate | 5e−4 | 1e−3 | 1e−3 | 5e−4 | 5e−4 | 3e−4 |
| Target weight | 1e−3 | 5e−3 | 5e−3 | 1e−3 | 1e−3 | 5e−3 |
| Batch size | 128 | 256 | 100 | 128 | 128 | 256 |
| Replay buffer size | 1e6 | 1e6 | 1e6 | 1e6 | 1e6 | 1e6 |
| Actor network architecture | 128, 128 | 400, 300 | 400, 300 | 128, 128 | 128, 128 | 256, 256 |
| Critic network architecture | 400, 300 | 400, 300 | 400, 300 | 400, 300 | 400, 300 | 256, 256 |
| Population size | 10 | 10 | 10 | 10 | × | × |
| Mutation probability | 0.9 | 0.9 | × | 0.9 | × | × |

* represents that there are multiple values used in algorithms. × represents that the parameters are not used in algorithms.

(2) CERL [23]: CERL utilizes a portfolio of policies that explore diverse regions of the solution space together. In this way, the algorithm is insensitive to the choice of hyperparameters. Besides, the computation resources are dynamically allocated to improve the efficiency of algorithm.
(3) PDERL [29]: PDERL proposes two novel genetic operators based on backpropagation, which are called proximal mutation and distillation crossover. These two operators avoid destructive influence and can improve the performance of ERL.
(4) CEM-RL [17]: CEM-RL utilizes CEM to enhance the search efficiency of hybrid methods. To make use of the advantages of CEM, CEM-RL applies a gradient step to half the population and does not use an extra RL actor.
(5) DDPG [19]: DDPG applies the idea of DQN to the continuous action space and is one of the most widely used policy gradient methods. Besides, DDPG is also the RL algorithm adopted in ERL and PDERL.
(6) TD3 [21]: TD3 addresses the problem of value function overestimation in DDPG and shows superior performance in a range of OpenAI gym tasks [34]. Besides, TD3 is generally considered a state-of-the-art off-policy RL algorithm.

### 4.2. Benchmark problems and performance metrics

Mujoco environments [33] are widely utilized in the field of RL [70–72], and are conveniently accessible through the OpenAI gym [34], which serves as an interface for researchers and engineers. In this study, we evaluate the performance of several RL algorithms on six robot locomotion problems that are powered by Mujoco, namely HalfCheetah, Swimmer, Hopper, Reacher, Ant, and Walker-2d, as depicted in Fig. 4. The objective of these problems is to apply torque to the joints of the robot and enable it to accomplish specific tasks. Table 1 provides some basic information regarding these environments [34].

In addition to evaluating these algorithms on Mujoco environments, we also apply them to four classic control problems to showcase their unique features. These four problems are BipedalWalker, LunarLanderContinuous, MountainCarContinuous, and Pendulum, as shown in Fig. 5. Basic information for each of these environments can be found in Table 1 [34].

The performance of the agents is evaluated based on the cumulative reward achieved in a single episode. In the case of ERL, CERL, and PDERL, the best individual in the population is selected for testing. For CEM-RL, an actor with network weights that represent the mean of the distribution is tested. The performance of the RL algorithms is evaluated every five thousand steps. All selected agents are tested for five episodes and the average score are recorded as the performance of algorithm. In order to ensure a fair comparison between the RL algorithms and population-based algorithms, the number of steps taken by every individual in the population is accumulated. Additionally, the final reported results are based on the average values from five independent runs that employ different random seeds.

### 4.3. Experimental settings

For all the compared ERLs and RL algorithms, we utilize the implementation provided by the respective authors. The parameter settings in each algorithm are set as suggested in their corresponding papers. The important hyperparameters used in the compared algorithms are listed in Table 2. The network architecture used in all compared algorithms is a multi-layer perceptron with 2 hidden layers. The corresponding hyperparameters in this table represent the number of neurons in hidden layers. For Mujoco environments, all algorithms undergo the same number of steps as in ERL [4], which is 2 million on HalfCheetah, Swimmer, and Reacher, 4 million on Hopper, 6 million on Ant, and 8 million on Walker2d. For four classic control environments, all algorithms are run for 1 million steps. To ensure a fair comparison, all experiments are conducted on the same software environments and hardware devices.

### 4.4. Experimental results and performance analysis

Fig. 6 presents the learning curves of the compared algorithms for solving six continuous control problems, where the shaded areas represent the standard variation in performance. Table 3 provides the mean

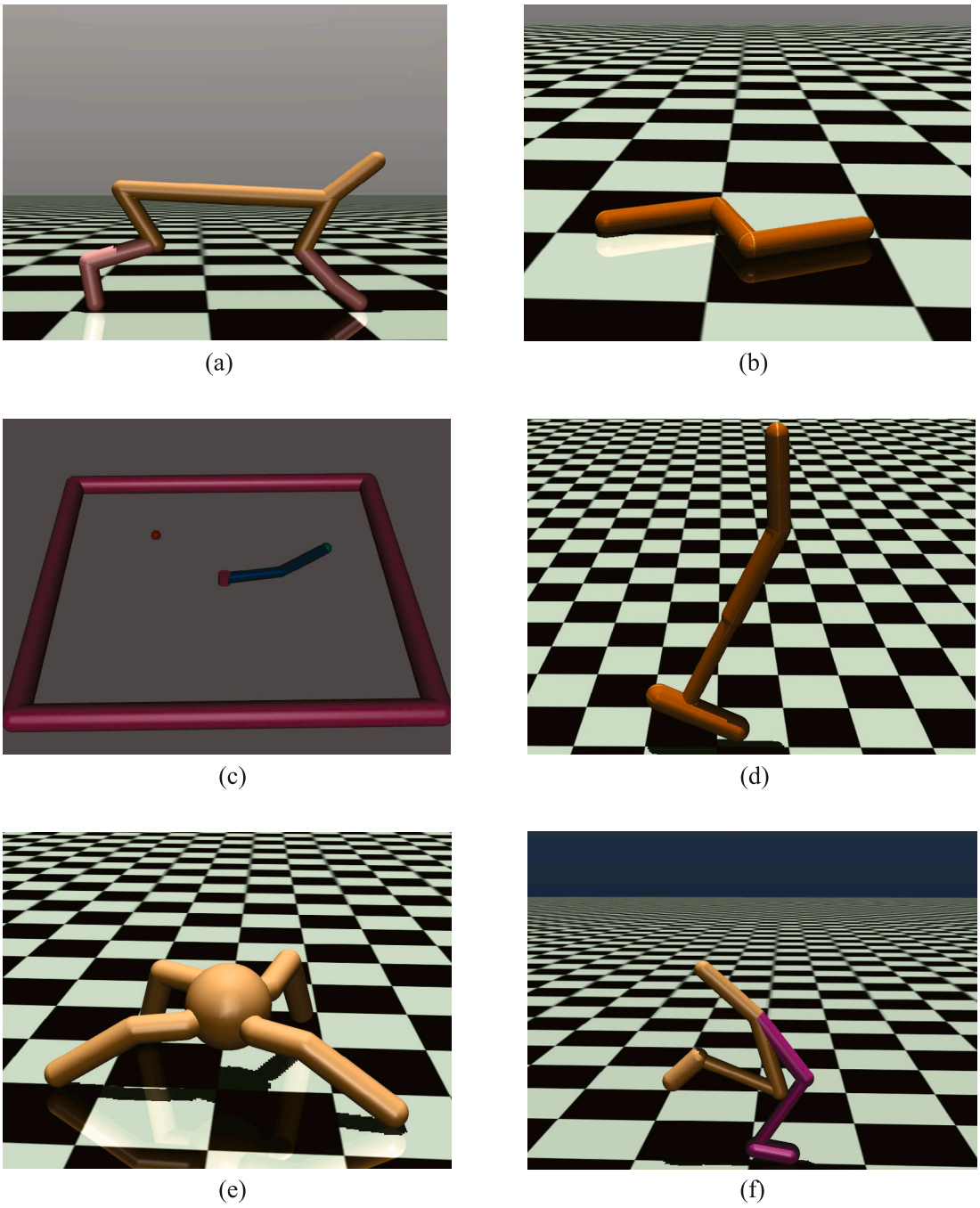



**Fig. 4.** Mujoco continuous control benchmarks. (a) HalfCheetah; (b) Swimmer; (c) Reacher; (d) Hopper; (e) Ant; (f) Walker-2d.

**Table 3**
The mean and standard variation of final performance in all Mujoco test environments.

| Environment | ERL | CERL | CEM-RL | PDERL | DDPG | TD3 |
|---|---|---|---|---|---|---|
| HalfCheetah | 8631 ± 108 | 6964 ± 2636 | 11921 ± 413 | 10493 ± 217 | 3074 ± 1505 | 11918 ± 1322 |
| Swimmer | 358 ± 4 | 175 ± 98 | 168 ± 105 | 347 ± 13 | 136 ± 11 | 132 ± 13 |
| Reacher | −4.3 ± 0.3 | −6.9 ± 0.4 | −5.1 ± 0.9 | −4.7 ± 0.3 | −4.4 ± 0.9 | −3.7 ± 0.6 |
| Hopper | 1850 ± 608 | 2818 ± 775 | 3832 ± 54 | 3546 ± 97 | 2117 ± 750 | 2900 ± 650 |
| Ant | 4996 ± 1991 | 5483 ± 2402 | 4939 ± 1587 | 6116 ± 1075 | 5344 ± 1676 | 6552 ± 419 |
| Walker2d | 1779 ± 917 | 3747 ± 302 | 4469 ± 419 | 5401 ± 669 | 2413 ± 870 | 5362 ± 203 |

and standard deviation of the final performance across different Mujoco test environments. Overall, PDERL, CEM-RL, and TD3 demonstrate the best performance across all tasks. Despite being one of the pioneering works, ERL still remains competitive in some environments and outperforms other compared algorithms on Swimmer. While CERL is competitive in some environments, it does not significantly outperform other algorithms in our experiments. This may be due to the fact that one of the main contributions of CERL is to alleviate the sensitivity of hyperparameters in RL algorithms, but the hyperparameters in the tested algorithms may have already been well-tuned. Furthermore,

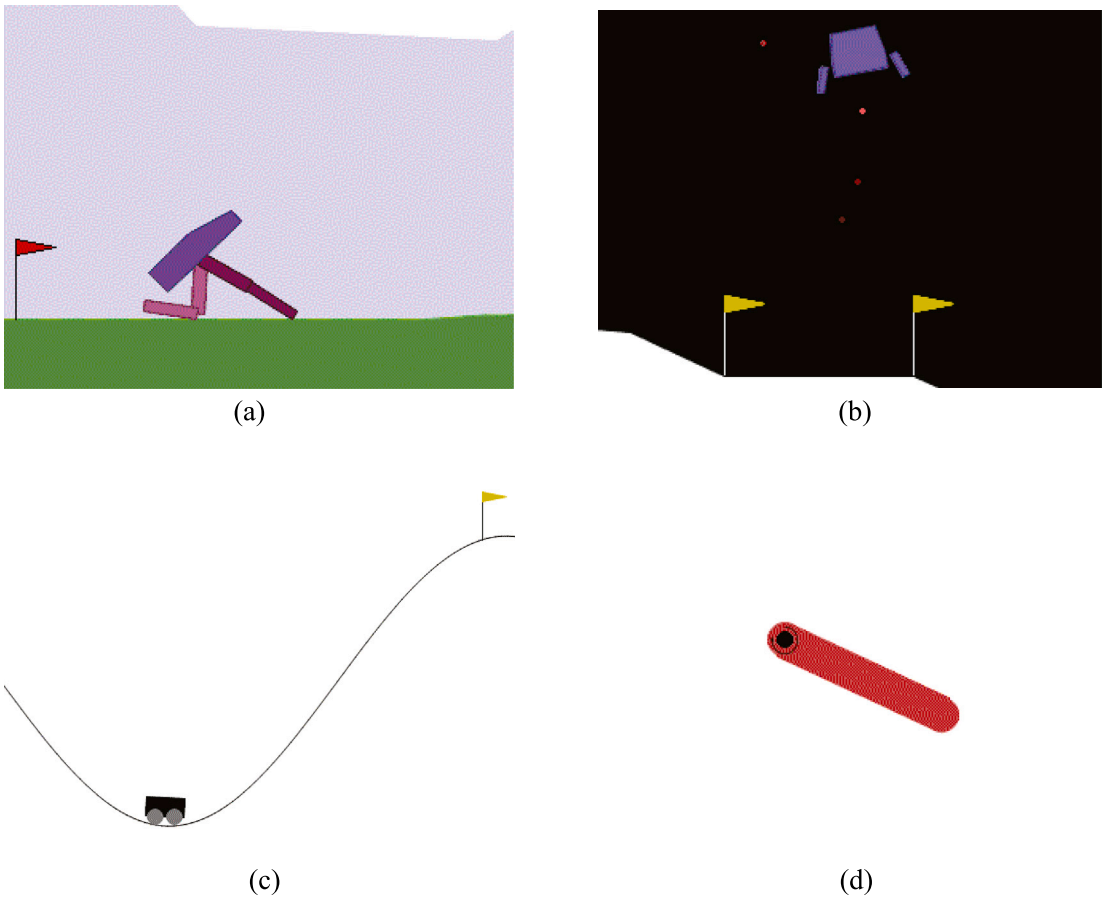



**Fig. 5.** Classic continuous control problems. (a) BipedalWalker; (b) LunarLanderContinuous; (c) MountainCarContinuous; (d) Pendulum.

**Table 4**
The mean and standard variation of final performance in classic control environments.

| Environment | ERL | CERL | CEM-RL | PDERL | DDPG | TD3 |
|---|---|---|---|---|---|---|
| BipedalWalker | $-25 \pm 32$ | $210 \pm 150$ | $293 \pm 11$ | $110 \pm 112$ | $53 \pm 143$ | $304 \pm 10$ |
| LunarLanderContinuous | $195 \pm 40$ | $142 \pm 170$ | $285 \pm 3$ | $248 \pm 17$ | $287 \pm 3$ | $275 \pm 6$ |
| MountainCarContinuous | $0 \pm 0$ | $57 \pm 46$ | $75 \pm 37$ | $94 \pm 1$ | $0 \pm 0$ | $0 \pm 0$ |
| Pendulum | $-581 \pm 178$ | $-154 \pm 8$ | $-142 \pm 4$ | $-449 \pm 158$ | $-163 \pm 10$ | $-153 \pm 25$ |

PDERL outperforms ERL in most environments, thereby validating the effectiveness of the genetic operators proposed in PDERL.

In the HalfCheetah environment, CEM-RL and TD3 demonstrate better performance than other algorithms, which may be attributed to the high efficiency of the TD3 algorithm. While CEM-RL exhibits faster learning speed than TD3 in the early stages, it is ultimately caught up in subsequent learning, as illustrated in Fig. 6. Although the performance of PDERL is not as good as that of these two algorithms, the RL algorithm used in PDERL is DDPG, which generally performs worse than TD3 in these continuous control tasks. Therefore, PDERL's performance may be further improved by replacing DDPG with TD3.

ERL and PDERL demonstrate superior performance on the Swimmer benchmark compared to other algorithms. The key advantage of these algorithms is their utilization of EAs within their framework. While RL algorithms typically struggle to learn on the Swimmer benchmark, EAs are better suited for this task as demonstrated in previous work [4]. This is due to the fact that EAs and RL methods explore distinct regions of the solution space. The experiments in [17] show that only CEM can also perform well on Swimmer. Therefore, the combination schemes of CEM-RL may limit the influence of CEM.

The Hopper and Walker2d environments are known to be unstable, resulting in large variations in agent performance across evaluation episodes. Therefore, it is crucial to protect high-performing individuals from being disrupted in these environments. TD3 and CEM-RL exhibit a noticeable drop in their learning curve, whereas PDERL's elitism and proximal mutation strategies result in more stable performance on these two benchmarks. For Reacher, most algorithms produce comparable results, with TD3 slightly outperforming the others. In Ant environment, CEM-RL's performance is not as strong as in other environments, but its stand variation in final performance is large. This suggests that CEM-RL can achieve high scores but lacks stability in Ant.

Fig. 7 depicts the learning curves of various algorithms compared on four classic continuous control problems. The shaded areas indicate the standard variation in performance. Table 4 presents the mean and standard deviation of final performance across different test problems. Overall, CEM-RL and TD3 achieve superior results on BipedalWalker, LunarLanderContinuous, and Pendulum. However, their performance is not as satisfactory on the MountainCarContinuous environment, which is due to the deceptive reward rule of this environment. It penalizes the agent by giving a negative reward when the agent executes a move action. Therefore, if the agent cannot reach the top of the hill to obtain the positive reward, it will tend to not move and receive a reward of zero. To overcome this problem, algorithms must enable enough exploration and allow the agent to reach the top of the hill. In this environment, ERLs, particularly PDERL, demonstrate better exploration ability and performance than other kinds of RL algorithms.

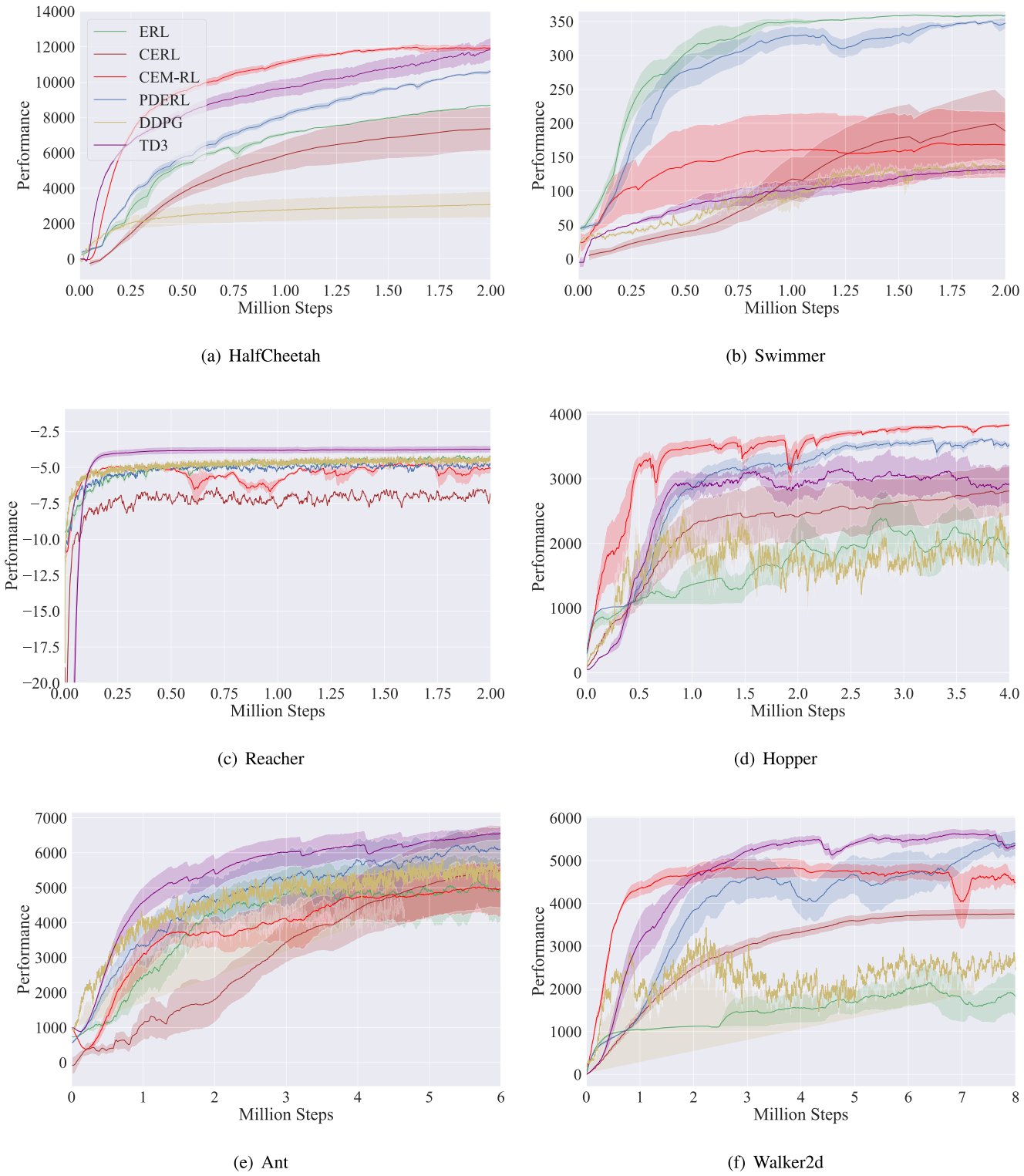
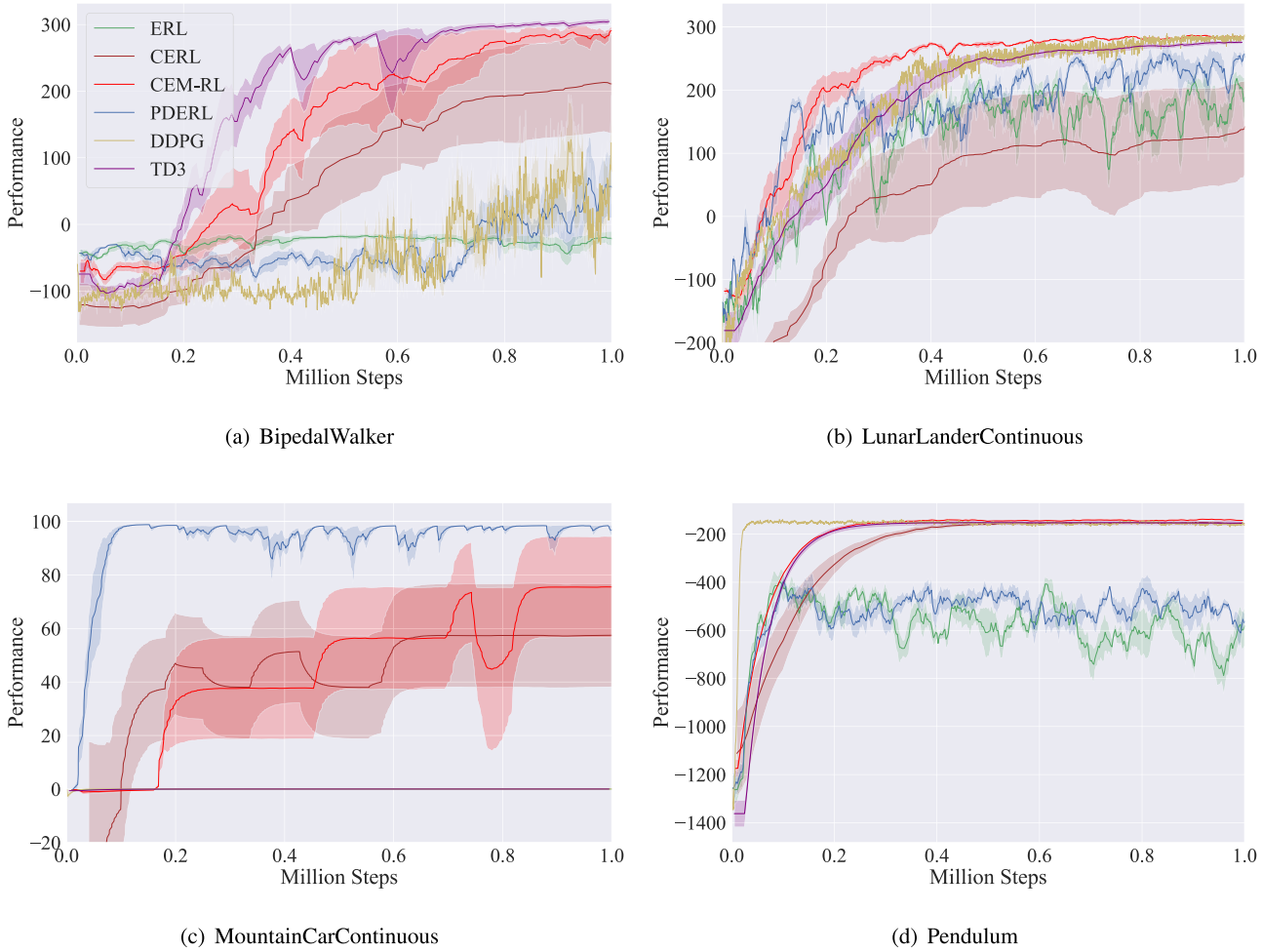



**Fig. 6.** Learning curves on six Mujoco continuous control benchmarks.

## 5. Conclusions

Over the past few years, numerous evolutionary reinforcement learning algorithms have been proposed and have gained considerable attention. These algorithms draw upon different techniques from both evolutionary computation and reinforcement learning or introduce novel combination mechanisms to improve their performance. In this paper, we provide a comprehensive survey of these ERLs. First, we introduce some background information and popular components in ERLs. We then classify ERLs based on the EAs used in their framework and the types of individuals in the population. We discuss and analyze the main ideas and advantages of these algorithms. Additionally, we conduct comparative experiments to examine the practical performance of some popular ERLs in a variety of continuous control tasks. The experimental results reveal that the different EAs in ERLs exhibit various features and performance. On these benchmarks, these algorithms




(a) BipedalWalker (b) LunarLanderContinuous

(c) MountainCarContinuous (d) Pendulum

**Fig. 7.** Learning curves on four classic continuous control problems.

demonstrate distinct stability and exploration ability. Furthermore, the combination mechanism can have a significant impact on the performance of ERLs.

Among the various combinations of EAs and RL, GA and ES are the most commonly used evolutionary methods. However, there are other EAs with different characteristics in evolutionary computation, such as Differential Evolution (DE) [73,74] and Ant Colony Optimization (ACO) [75], which could be integrated into ERL frameworks to explore new directions for future work. Additionally, most ERLs are built on off-policy RL algorithms such as DDPG, TD3, or SAC. Only a few hybrid algorithms leverage EAs to enhance the performance of on-policy or value-based RL algorithms. Therefore, there is ample room for improvement in ERL studies based on these RL algorithms. Furthermore, nearly all of the ERLs investigated in the literature employ simple feedforward neural networks, such as multilayer perceptrons [76]. As a result, exploring more appropriate models for ERLs represents another promising direction for future research [77–81].

## CRediT authorship contribution statement

**Qingling Zhu:** Conceptualization, Methodology, Software, Writing. **Xiaoqiang Wu:** Conceptualization, Validation, Writing – review & editing. **Qiuzhen Lin:** Supervision, Validation, Writing – review & editing, Project administration. **Lijia Ma:** Writing – review & editing. **Jianqiang Li:** Supervision, Project administration. **Zhong Ming:** Supervision, Project administration. **Jianyong Chen:** Supervision, Project administration.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62203308 and 62173236, in part by the China Postdoctoral Science Foundation under Grant 2022M712070, in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076, in part by Natural Science Foundation of Guangdong Province under Grant 2023A1515011238, in part by the Shenzhen Science and Technology Program under Grants JCYJ20220531101411027, JCYJ20190808174801673 and JCYJ20190808164211203, in part by the Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08X603, in part by the Guangdong "Pearl River Talent Plan" under Grant 2019JC01X235, and in part by the Shenzhen Science and Technology Innovation Commission under Grant R2020A045.

## References


[1] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354–359.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al., Dota 2 with large scale deep reinforcement learning, 2019, arXiv preprint arXiv:1912.06680.

[4] S. Khadka, K. Tumer, Evolution-guided policy gradient in reinforcement learning, Adv. Neural Inf. Process. Syst. 31 (2018).

[5] D. Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (2) (1994) 65–85.

[6] H.-G. Beyer, H.-P. Schwefel, Evolution strategies–A comprehensive introduction, Natural Comput. 1 (1) (2002) 3–52.

[7] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intell. 1 (1) (2007) 33–57.

[8] W.B. Langdon, R. Poli, N.F. McPhee, J.R. Koza, Genetic programming: An introduction and tutorial, with a survey of techniques and applications, in: Computational Intelligence: A Compendium, Springer, 2008, pp. 927–1028.

[9] X. Zhou, A.K. Qin, M. Gong, K.C. Tan, A survey on evolutionary construction of deep neural networks, IEEE Trans. Evol. Comput. 25 (5) (2021) 894–912.

[10] Z.-L. Sun, D.-S. Huang, C.-H. Zheng, L. Shang, Optimal selection of time lags for TDSEP based on genetic algorithm, Neurocomputing 69 (7–9) (2006) 884–887.

[11] B.L. Miller, D.E. Goldberg, et al., Genetic algorithms, tournament selection, and the effects of noise, Complex Syst. 9 (3) (1995) 193–212.

[12] L. Xie, A. Yuille, Genetic cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1379–1388.

[13] A.E. Eiben, J.E. Smith, et al., Introduction to Evolutionary Computing, Vol. 53, Springer, 2003.

[14] C.W. Ahn, R.S. Ramakrishna, Elitism-based compact genetic algorithms, IEEE Trans. Evol. Comput. 7 (4) (2003) 367–385.

[15] R.Y. Rubinstein, D.P. Kroese, The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning, Vol. 133, Springer, 2004.

[16] N. Hansen, The CMA evolution strategy: A tutorial, 2016, arXiv preprint arXiv:1604.00772.

[17] A. Pourchot, O. Sigaud, CEM-RL: Combining evolutionary and gradient-based methods for policy search, in: 7th International Conference on Learning Representations, ICLR, 2019.

[18] J. Huang, L. Huang, M. Liu, H. Li, Q. Tan, X. Ma, J. Cui, D.-S. Huang, Deep reinforcement learning-based trajectory pricing on ride-hailing platforms, ACM Trans. Intell. Syst. Technol. 13 (3) (2022) 1–19.

[19] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[20] F. Han, D.-S. Huang, A new constrained learning algorithm for function approximation by encoding a priori information into feedforward neural networks, Neural Comput. Appl. 17 (2008) 433–439.

[21] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.

[22] F.P. Such, V. Madhavan, E. Conti, J. Lehman, K.O. Stanley, J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, 2017, arXiv preprint arXiv:1712.06567.

[23] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, K. Tumer, Collaborative evolutionary reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 3341–3350.

[24] J. Hao, P. Li, H. Tang, Y. Zheng, X. Fu, Z. Meng, Erl-re$^2$: efficient evolutionary reinforcement learning with shared state representation and individual policy representation, International Conference on Learning Representations (2023).

[25] S. Majumdar, S. Khadka, S. Miret, S. McAleer, K. Tumer, Evolutionary reinforcement learning for sample-efficient multiagent coordination, in: International Conference on Machine Learning, PMLR, 2020, pp. 6651–6660.

[26] R. Lowe, Y.I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Adv. Neural Inf. Process. Syst. 30 (2017).

[27] E. Sachdeva, S. Khadka, S. Majumdar, K. Tumer, MAEDyS: multiagent evolution via dynamic skill selection, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2021, pp. 163–171.

[28] P. Li, J. Hao, H. Tang, Y. Zheng, X. Fu, Race: improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution, in: International Conference on Machine Learning, PMLR, 2023, pp. 19490–19503.

[29] C. Bodnar, B. Day, P. Lió, Proximal distilled evolutionary reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3283–3290.

[30] J. Lehman, J. Chen, J. Clune, K.O. Stanley, Safe mutations for deep and recurrent neural networks through output gradients, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2018, pp. 117–124.

[31] T. Osa, J. Pajarinen, G. Neumann, J.A. Bagnell, P. Abbeel, J. Peters, et al., An algorithmic perspective on imitation learning, Found. Trends® Robotics 7 (1–2) (2018) 1–179.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.

[33] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.

[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016, arXiv preprint arXiv:1606.01540.

[35] Y. Wang, T. Zhang, Y. Chang, B. Liang, X. Wang, B. Yuan, A surrogate-assisted controller for expensive evolutionary reinforcement learning, 2022, arXiv preprint arXiv:2201.00129.

[36] X. Wu, Q. Zhu, Q. Lin, J. Li, J. Chen, Z. Ming, An efficient evaluation mechanism for evolutionary reinforcement learning, in: International Conference on Intelligent Computing, Springer, 2022, pp. 41–50.

[37] H. Zheng, J. Jiang, P. Wei, G. Long, C. Zhang, Competitive and cooperative heterogeneous deep reinforcement learning, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2020, pp. 1656–1664.

[38] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.

[39] E. Marchesini, D. Corsi, A. Farinelli, Genetic soft updates for policy evolution in deep reinforcement learning, in: International Conference on Learning Representations, 2020.

[40] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, D. Silver, Rainbow: Combining improvements in deep reinforcement learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 3215–3222.

[41] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, (1) 2016, pp. 2094–2100.

[42] S. Chang, J. Yang, J. Choi, N. Kwak, Genetic-gated networks for deep reinforcement learning, Adv. Neural Inf. Process. Syst. 31 (2018).

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[44] Y. Wu, E. Mansimov, R.B. Grosse, S. Liao, J. Ba, Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation, Adv. Neural Inf. Process. Syst. 30 (2017).

[45] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K. Stanley, J. Clune, Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents, Adv. Neural Inf. Process. Syst. 31 (2018).

[46] J.K. Pugh, L.B. Soros, K.O. Stanley, Quality diversity: A new frontier for evolutionary computation, Front. Robotics AI (2016) 40.

[47] L. Shi, S. Li, Q. Zheng, M. Yao, G. Pan, Efficient novelty search through deep reinforcement learning, IEEE Access 8 (2020) 128809–128818.

[48] O. Nilsson, A. Cully, Policy gradient assisted map-elites, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2021, pp. 866–875.

[49] J.-B. Mouret, J. Clune, Illuminating search spaces by mapping elites, 2015, arXiv preprint arXiv:1504.04909.

[50] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer Science & Business Media, 2006.

[51] T. Pierrot, V. Macé, F. Chalumeau, A. Flajolet, G. Cideron, K. Beguir, A. Cully, O. Sigaud, N. Perrin-Gilbert, Diversity policy gradient for sample efficient quality-diversity optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 1075–1083.

[52] J.K. Franke, G. Köhler, A. Biedenkapp, F. Hutter, Sample-efficient automated deep reinforcement learning, 2020, arXiv preprint arXiv:2009.01555.

[53] M. Jaderberg, V. Dalibard, S. Osindero, W.M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al., Population based training of neural networks, 2017, arXiv preprint arXiv:1711.09846.

[54] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, 2017, arXiv preprint arXiv:1703.03864.

[55] Y. Tang, Guiding evolutionary strategies with off-policy actor-critic., in: AAMAS, 2021, pp. 1317–1325.

[56] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, N. de Freitas, Sample efficient actor-critic with experience replay, 2016, arXiv preprint arXiv:1611.01224.

[57] K. Lee, B.-U. Lee, U. Shin, I.S. Kweon, An efficient asynchronous method for integrating evolutionary and gradient-based policy search, Adv. Neural Inf. Process. Syst. 33 (2020) 10124–10135.

[58] J. Liu, L. Feng, Diversity evolutionary policy deep reinforcement learning, Comput. Intell. Neurosci. 2021 (2021).

[59] K. Suri, X.Q. Shi, K.N. Plataniotis, Y.A. Lawryshyn, Maximum mutation reinforcement learning for scalable control, 2020, arXiv preprint arXiv:2007.13690.
[60] H. Zheng, P. Wei, J. Jiang, G. Long, Q. Lu, C. Zhang, Cooperative heterogeneous deep reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 17455–17465.
[61] R. Simmons-Edler, B. Eisner, E. Mitchell, S. Seung, D. Lee, Q-learning for continuous actions with cross-entropy guided policies, 2019, arXiv preprint arXiv:1903.10605.
[62] Z. Shi, S.P. Singh, Soft actor-critic with cross-entropy policy optimization, 2021, arXiv preprint arXiv:2112.11115.
[63] L. Shao, Y. You, M. Yan, S. Yuan, Q. Sun, J. Bohg, Grac: Self-guided and self-regularized actor-critic, in: Conference on Robot Learning, PMLR, 2022, pp. 267–276.
[64] F. Han, Q.-H. Ling, D.-S. Huang, An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks, Neural Comput. Appl. 19 (2010) 255–261.
[65] Y. Ma, T. Liu, B. Wei, Y. Liu, K. Xu, W. Li, Evolutionary action selection for gradient-based policy learning, 2022, arXiv preprint arXiv:2201.04286.
[66] A. Hallawa, T. Born, A. Schmeink, G. Dartmann, A. Peine, L. Martin, G. Iacca, A. Eiben, G. Ascheid, Evo-RL: evolutionary-driven reinforcement learning, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 153–154.
[67] M. Colledanchise, P. Ögren, Behavior Trees in Robotics and AI: An Introduction, CRC Press, 2018.
[68] L. Pan, Q. Cai, L. Huang, Multi-path policy optimization, 2019, arXiv preprint arXiv:1911.04207.
[69] W. Jung, G. Park, Y. Sung, Population-guided parallel policy search for reinforcement learning, 2020, arXiv preprint arXiv:2001.02907.
[70] Y. Duan, X. Chen, R. Houthooft, J. Schulman, P. Abbeel, Benchmarking deep reinforcement learning for continuous control, in: International Conference on Machine Learning, PMLR, 2016, pp. 1329–1338.
[71] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018, pp. 3207–3214.
[72] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
[73] K.V. Price, Differential evolution, in: Handbook of Optimization, Springer, 2013, pp. 187–214.
[74] J.-X. Du, D.-S. Huang, X.-F. Wang, X. Gu, Shape recognition based on neural networks trained by differential evolution algorithm, Neurocomputing 70 (4–6) (2007) 896–903.
[75] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39.
[76] D.-S. Huang, A constructive approach for finding arbitrary roots of polynomials by neural networks, IEEE Trans. Neural Netw. 15 (2) (2004) 477–491.
[77] D. Wu, S.-J. Zheng, X.-P. Zhang, C.-A. Yuan, F. Cheng, Y. Zhao, Y.-J. Lin, Z.-Q. Zhao, Y.-L. Jiang, D.-S. Huang, Deep learning-based methods for person re-identification: A comprehensive review, Neurocomputing 337 (2019) 354–371.
[78] D.-S. Huang, S.-D. Ma, Linear and nonlinear feedforward neural network classifiers: A comprehensive understanding, J. Intell. Syst. 9 (1) (1999) 1–38.
[79] H. Li, X. Li, L. Su, D. Jin, J. Huang, D. Huang, Deep spatio-temporal adaptive 3d convolutional neural networks for traffic flow prediction, ACM Trans. Intell. Syst. Technol. 13 (2) (2022) 1–21.
[80] D. Wu, C. Wang, Y. Wu, Q.-C. Wang, D.-S. Huang, Attention deep model with multi-scale deep supervision for person re-identification, IEEE Trans. Emerg. Top. Comput. Intell. 5 (1) (2021) 70–78.
[81] D.-S. Huang, J.-X. Du, A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, IEEE Trans. Neural Netw. 19 (12) (2008) 2099–2115.