Evolutionary Reinforcement Learning via Cooperative Coevolution

Chengpeng Hu[a,b], **Jialin Liu**[a,b] **and Xin Yao**[c]

[a]Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
[b]Research Institute of Trustworthy Autonomous System, Southern University of Science and Technology, Shenzhen, China
[c]Department of Computing and Decision Sciences, Lingnan University, Hong Kong SAR, China

Abstract. Recently, evolutionary reinforcement learning has obtained much attention in various domains. Maintaining a population of actors, evolutionary reinforcement learning utilises the collected experiences to improve the behaviour policy through efficient exploration. However, the poor scalability of genetic operators limits the efficiency of optimising high-dimensional neural networks. To address this issue, this paper proposes a novel cooperative coevolutionary reinforcement learning (CoERL) algorithm. Inspired by cooperative coevolution, CoERL periodically and adaptively decomposes the policy optimisation problem into multiple subproblems and evolves a population of neural networks for each of the subproblems. Instead of using genetic operators, CoERL directly searches for partial gradients to update the policy. Updating policy with partial gradients maintains consistency between the behaviour spaces of parents and offspring across generations. The experiences collected by the population are then used to improve the entire policy, which enhances the sampling efficiency. Experiments on six benchmark locomotion tasks demonstrate that CoERL outperforms seven state-of-the-art algorithms and baselines. Ablation study verifies the unique contribution of CoERL's core ingredients.

1 Introduction

Reinforcement learning (RL) has made adequate advancements in various domains such as video games [20, 33], Go [29] and robotic control [10] with a competitive level beyond human. Evolutionary reinforcement learning (ERL) [11, 1] combines evolutionary computation and RL to solve sequential decision-making problems, capitalising on the benefits of evolution in exploration. ERL maintains a population of parameterised actors, called individuals, which interact with the paralleled environments to collect experiences. Individuals with higher reward-based fitness are more likely to be selected as the parents, to which mutation and crossover operators are applied to reproduce the next generation. Another RL agent called *learner*, learns from diverse experiences sampled by the maintained population of actors and substitutes the individual with the worst fitness in the population periodically, i.e., injecting its gradient information for better convergence.

However, ERL introduces a scalability problem, attributed to the genetic operators used for reproducing new individuals [43, 26, 14]. Minor perturbations in the parameter space, caused by genetic operators, lead to significant divergences in behaviour space [13, 14]. Consequently, an offspring might not inherit its parents' behaviours by merely exchanging fragments of parameters. Thus, this inconsistency between parameter space and behaviour space does not align with the principles of evolution [42].

Lehman et al. [14] proposed a safe mutation operator based on the sensitivity of neural networks' outputs, which slightly adjusts the parameters of neural networks to ensure consistency. State-based crossover [7], as well as distillation crossover and proximal mutation operator [2], improve the traditional genetic operators using back-propagation. However, those operators are gradient-based, which introduce in-negligible extra computational costs. Nevertheless, the works of [22] and [39] directly searched for the parameters of neural networks using the cooperative coevolution, in which the policy optimisation problem is decomposed into multiple low-dimensional subproblems. However, the experiences of optimising subproblems are not fully utilised, resulting in inefficient training [11].

To address the scalability issue, we propose a novel cooperative coevolutionary reinforcement learning (CoERL) algorithm. The optimisation of a high-dimensional neural network is decomposed into multiple low-dimensional subproblems periodically and adaptively by cooperative coevolution. A population is resampled for each subproblem, in which each individual maintains the same neural network architecture. CoERL directly searches for partial gradients and updates the entire policy. The partial gradient searched via the population guides a proximal update on the subproblem, diminishing inconsistency between the behaviour spaces of parents and offspring. Experiences collected during evolution are then used to enhance RL for better sampling efficiency.

Our main contributions are summarised as follows:

- This paper proposes CoERL, a novel cooperative coevolutionary reinforcement learning algorithm to address the scalability issue of ERL, which also improves sample efficiency during training.
- This paper proposes a partially updating strategy for policy improvement based on cooperative coevolution, which costs less and maintains the consistency between the behaviour spaces of parents and offspring.
- Experimental results on six benchmark locomotion tasks demonstrate the comparable performance of CoERL, compared with seven state-of-the-art algorithms and baselines. Ablation study

shows the unique contributions of CoERL's core ingredients.

2 Background

In this section, we introduce some preliminary, cooperative coevolution and recent progress of ERL.

2.1 *Preliminary: Markov decision process*

Markov decision process (MDP) [30] is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition probability function, and γ is the discount factor. A policy π is a mapping from states to probability distributions for acting, where $\pi(a_t|s_t)$ is the probability of taking action a_t in state s_t at time t. The goal of the MDP is to optimise a parameterised policy π_θ that maximises the discounted cumulative reward, formulated as:

$$\max_{\theta} \; J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})], \tag{1}$$

where s_0 is an initial state. $\tau \sim \pi_\theta$ denotes a trajectory $(s_0, a_0, s_1, a_1, \ldots, s_t, a_t, s_{t+1})$ sampled from π_θ.

2.2 *Cooperative coevolution*

Cooperative coevolution (CC) simulates the interactions of species living in an ecosphere [18]. Potter and De Jong [24] proposed the first CC algorithm, called cooperative coevolutionary genetic algorithm (CCGA). CCGA decomposes an optimisation problem into multiple subproblems. A population is initialised for each subproblem with a context vector for fitness evaluations. For example, given an optimisation problem with n variables, the fitness of a partial solution x_{ij}, i.e., the j-th individual in the population of i-th subproblem, can be evaluated by replacing the variable b_j in the context vector $\boldsymbol{b}$. Each variable in $\boldsymbol{b}$ is updated with the best individual for the corresponding subproblem.

Problem decomposition referring to the grouping of variables is critical in CC [18]. Incorrect grouping leads to a local optimum, which is limited by the decomposed problem itself [23, 38]. Intuitively, static grouping and random grouping [24] are easy to implement and require no additional computational budget or knowledge. However, the effectiveness of the general strategies is unclear and requires some presets. Interaction-based grouping [16] and landscape-aware grouping [37] decompose the problem according to the characteristics of variables such as correlation and landscape. However, additional costs for fitness evaluations are required.

Another issue concerns the evaluation of individuals within a subproblem's population, also known as the credit assignment problem, since individuals in a population represent only part of the entire solution. Collaboration for fitness evaluation with other subproblems' populations is required. A global context vector can be constructed by collaborators (i.e., individuals) selected from populations of subproblems. Popular selection methods, such as single best collaborator [24], elite collaborator [8], and random collaborator selection, are typically chosen based on the specific domain [18].

Cooperative coevolution, served as a useful technique for solving high-dimensional black-box optimisation problems, is expected to benefit the policy optimisation. In this paper, we apply random grouping and single best collaboration for low cost and simplicity.

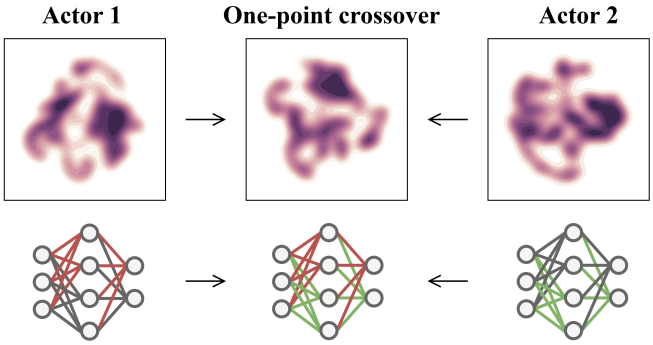

Figure 1. Inconsistency phenomenon between the behaviour spaces of parents and offspring by applying one-point crossover to exchange partially the parameters of Actor 1 (red) and Actor 2 (green). Feature maps show the behaviour spaces decompressed by t-SNE [32].

2.3 *Evolutionary reinforcement learning*

Evolutionary algorithms (EAs) have been successfully applied to solve RL problems, where neuroevolution and evolution-guided policy gradient are two main trends in this area [21, 43, 36, 28, 1].

2.3.1 *Neuroevolution*

Neuroevolution directly optimises the parameters or architectures of neural networks without considering the underlying mechanism like gradients, making it suitable for evolving reinforcement learning policy and tackling real-world problems [17, 4, 15, 6, 40]. Fitness metrics formulated on rewards are used to select good individuals for reproducing the next generation by mutation and crossover. Hence, neuroevolution can intuitively handle complicated rewards such as non-convex and non-differentiable cases [26, 4]. Population-based training also encourages exploration. However, Nesterov and Spokoiny [22] indicated that EAs scale poorly with the increase of parameters.

Gong et al. [9] combined coevolution and backpropagation to solve classification tasks, while the framework is limited by traditional reproduction and specific domains. Yang et al. [39] proposed a CC framework based on negatively correlated search (NCS) to tackle this issue, which directly searches for the parameters of a neural network. However, it only utilises the final accumulative reward as fitness, resulting in a waste of experiences collected during evolution.

Besides, traditional genetic operators limit EA's capability of addressing large-scale optimisation problems. The direct operations on the parameter space lead to an inconsistency between the behaviour spaces of parents and offspring, where parental behaviours may be typically forgotten [14, 7, 2]. Figure 1 shows an example of applying the one-point crossover, in which behaviour spaces of actors are decompressed by t-SNE [32]. It is easy to see from Figure 1 that directly exchanging the parameter fragments of the parents leads to a forgetting phenomenon in the behaviour space of the offspring.

2.3.2 *Evolution-guided policy gradient*

Evolution-guided policy gradient combines neuroevolution and MDP-based RL [11]. In the EA loop, actors, also called individuals, interact with the environments and are evolved by genetic operators. Experiences collected during the evaluations, i.e., transitions, are stored in a replay buffer and leveraged in the RL loop. Khadka et al. [12] proposed a portfolio of policies and dynamically allocated computational resources to train different policies, extending

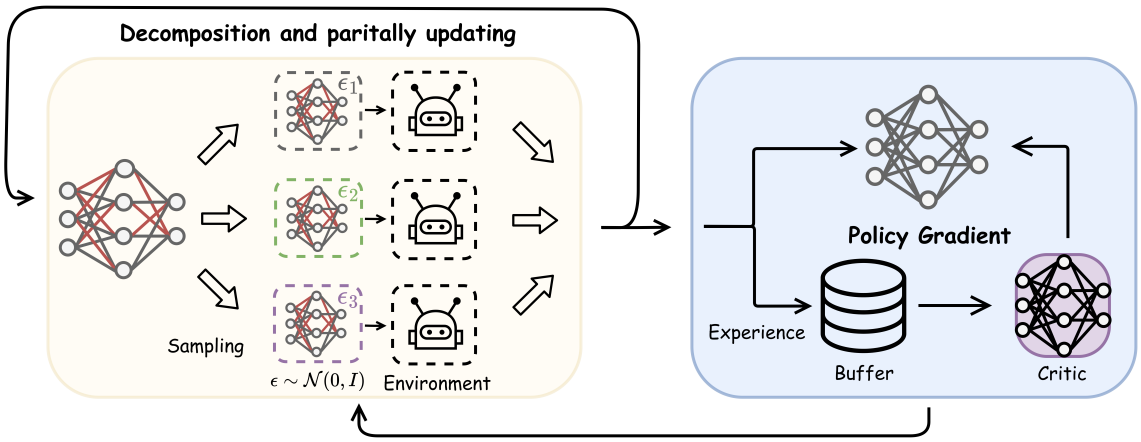

Figure 2. Diagram of CoERL. The CoERL algorithm begins by decomposing the policy optimisation problem parameterised by a neural network into multiple subproblems and searching for partial gradients to update the policy. Subsequently, the explored experiences are gathered to further refine the policy outcome using the MDP-based RL.

ERL. CEM-RL [25] leveraged cross-entropy method (CEM) to delayed deep deterministic policy gradient (TD3). Without using genetic operators, CEM-RL samples actors from the policy distribution estimated in previous generations. Gangwani and Peng [7] applied imitation learning to the crossover operator, which use trajectories of two parents to train an offspring. Bodnar et al. [2] improved the mutation and crossover by using local replay memory and the critic of the RL learner, namely proximal distilled evolutionary reinforcement learning (PDERL). New offsprings are produced by exchanging the local replay buffer or perturbation according to the sensitivity of the actions in PDERL [2]. Genetic algorithm has been combined with RL in the work of [19], where individuals are generated by the noisy mutation only. To tackle the computationally expensive evaluation, Wang et al. [34] extended the surrogate model to ERL, which estimates the fitness of each individual based on the approximated value function. Zhu et al. [44] addressed the exploration and exploitation issue by maintaining actor and critic populations.

Although ERL inherits the advantages of neuroevolution, enabling it to address the temporal credit assignment problem with sparse rewards, it faces the challenge of poor scalability with EAs. Recent efforts have primarily concentrated on enhancing genetic operators, but at the expense of increased computational costs.

3 Cooperative coevolutionary reinforcement learning

Aiming at addressing the poor scalability of ERL, we propose a novel cooperative coevolutionary reinforcement learning (CoERL) algorithm. Pseudo-code and diagram of CoERL are provided in Algorithm 1 and Figure 2, respectively.

3.1 Collaboration between cooperative coevolution and reinforcement learning

CoERL maintains collaborative loops between CC and RL. In the CC loop, a neural network (a policy) is converted into a position-fixed vector with real numbers. The policy optimisation problem, is divided into multiple subproblems, in which parameters of the neural network are grouped. The decomposed subproblems have no intersection, while their union collectively constitute the entire parameter space. Given a policy parameterised by θ, it ensures that $\theta^{\langle \mathcal{I}_i \rangle} \cap \theta^{\langle \mathcal{I}_j \rangle} = \varnothing, \forall i \neq j, 1 \leq i \leq m, 1 \leq j \leq m$ and $\theta^{\langle \mathcal{I}_1 \rangle} \cup \theta^{\langle \mathcal{I}_2 \rangle} \cup \cdots \cup \theta^{\langle \mathcal{I}_m \rangle} = \theta$, where m is the number of subproblems and $\langle \mathcal{I}_i \rangle$ denotes the parameter indices of subproblem j. To optimise a subproblem, a population is maintained and generated using perturbation methods instead of traditional genetic operators. Practically, each individual in the population shares the same neural network structure with the freezing quotient set of the subproblem. Only the variables in the subproblem, which form part of the neural network, are perturbed by the noises from the distribution.

Individuals consistently interact with the environment. The cumulative reward obtained by the individual is treated as its fitness. Then, a number of noised individuals and their corresponding fitness are used to update the entire policy. Notably, subproblems are not optimised separately. The policy optimised according to the preceding subproblem is regarded as the complementary base of the subsequent subproblem. Thus, connections between different subproblems are built.

After updating the policy within each subproblem, CoERL proceeds to optimise the policy via MDP-based RL. Experiences such as transitions produced by the individuals during the cooperative coevolution loop are collected in a replay buffer. Batches sampled from the replay buffer are then used to update the policy via policy gradient, which fully utilises the temporal information. Sections 3.2 and 3.3 detail the two loops, respectively.

3.2 Partially updating via cooperative coevolution

Given a policy π_θ parameterised by θ, CoERL decomposes the policy optimisation problem into m subproblems in the CC loop. The subproblem j of policy optimisation, $\theta^{\langle \mathcal{I}_j \rangle}$ indexed by parameters indices $\langle \mathcal{I}_j \rangle$, and the quotient $\hat{\theta}^{\langle \mathcal{I}_j \rangle} = \theta / \theta^{\langle \mathcal{I}_j \rangle}$ constitute the entire parameter space, i.e., $\theta = \left[\theta^{\langle \mathcal{I}_j \rangle} : \hat{\theta}^{\langle \mathcal{I}_j \rangle}\right]$. Then, for each subproblem j, a corresponding actor $\pi_{\theta^{\langle \mathcal{I}_j \rangle}}$ is constructed by freezing the quotient part. CoERL partially updates the policy by iteratively optimising each subproblem. A population is first sampled via a specific distribution $P^{\langle \mathcal{I}_j \rangle}$. Each individual is an actor $\pi_{\theta^{\langle \mathcal{I}_j \rangle}}$, where $\theta^{\langle \mathcal{I}_j \rangle} \sim P^{\langle \mathcal{I}_j \rangle}$. All individuals in the population of the subproblem share the same quotient part $\hat{\theta}^{\langle \mathcal{I}_j \rangle}$. Then, the fitness of each individual $f(\pi_{\theta^{\langle \mathcal{I}_j \rangle}})$ is determined by the cumulative reward. The expected

Algorithm 1 CoERL.

Require: The number of generations T, population size μ, noise strength σ, number of subproblems m, learning rate α, temperature coefficient α_s

Ensure: Policy π_θ

Initialise policy π_θ, two critics $\hat{Q}_{\phi_1}$ and $\hat{Q}_{\phi_2}$
Initialise reward buffer $\mathcal{B}_\mathcal{R}$
for $n = 1$ to T **do**
  Decompose the policy optimisation problem into m sub-problems by grouping $|\theta|$-d parameters to m sets of parameter indices $\mathcal{I}_1, \cdots, \mathcal{I}_m$
  for $j = 1$ to m **do**
    Sample $\epsilon_1, \cdots, \epsilon_\mu \sim \mathcal{N}^{|\mathcal{I}_j|}(0, I)$
    Reproduce a population $\{\psi_1, \cdots, \psi_\mu\}$ for sub-problem $\mathcal{I}_j$ by $\psi_i \leftarrow \theta^{\langle \mathcal{I}_j \rangle} + \epsilon_i$, for $i = 1, \cdots, \mu$
      ▷ $\theta^{\langle \mathcal{I}_j \rangle}$ *denotes θ's elements at indices $\mathcal{I}_j$*
    for $i = 1$ to μ **do**
      $J_\mathcal{R}^{\pi_{\psi_i}}, \tau^{\pi_{\psi_i}} = \text{Evaluate}(\pi_{\psi_i})$
      Store $\tau^{\pi_{\psi_i}}$ in $\mathcal{B}_\mathcal{R}$
      Assign fitness $f_i = J_\mathcal{R}^{\pi_{\psi_i}}$
    end for
    $\theta^{\langle \mathcal{I}_j \rangle} \leftarrow \theta^{\langle \mathcal{I}_j \rangle} + \alpha \frac{1}{\mu\sigma} \sum_{i=1}^{\mu} f_i \epsilon_i$ ▷ *Eq.* (5)
      ▷ *Update parameters of policy π_θ indexed by $\mathcal{I}_j$*
  end for
  ▷ *Policy gradient with $\mathcal{B}_\mathcal{R}$:*
  Randomly sample a minibatch $\mathcal{B}$ of transitions $\mathcal{T} = \langle s, a, s', r \rangle$ from $\mathcal{B}_R$
  Compute target $y = r - \gamma(\min_{j=1,2} \hat{Q}_{\phi_j}(s', \tilde{a}') - \alpha_s \log \pi_\theta(\tilde{a}'|s'))$, where $\tilde{a}' \sim \pi_\theta(\cdot|s')$
  Update critics with $\nabla_{\phi_j} \frac{1}{|B_R|} \sum_{\mathcal{T} \in \mathcal{B}} (y - \hat{Q}_{\phi_j}(s, a))^2$ for $j = 1, 2$
  Update actor with $\nabla_\theta \frac{1}{|B_R|} \sum_{\mathcal{T} \in \mathcal{B}} (\min_{j=1,2} \hat{Q}_{\phi_j}(s, \tilde{a}_\theta) - \alpha_s \log \pi_\theta(\tilde{a}_\theta|s))$, where $\tilde{a}_\theta$ is sampled from $\pi_\theta(\cdot|s)$ via the reparameterisation trick
end for

fitness under an arbitrary distribution is formulated as Eq. (2):

$$J(\theta^{\langle \mathcal{I}_j \rangle}) = \mathbb{E}_{\theta^{\langle \mathcal{I}_j \rangle}}[f(\pi_{\theta^{\langle \mathcal{I}_j \rangle}})] = \int f(\pi_{\theta^{\langle \mathcal{I}_j \rangle}}) p(\theta^{\langle \mathcal{I}_j \rangle}) d\theta^{\langle \mathcal{I}_j \rangle}, \tag{2}$$

where $p(\theta^{\langle \mathcal{I}_j \rangle})$ denotes the density. Then, we can write the gradient form using the "log-likelihood trick". The estimation of the gradient by maintaining a population with size μ is shown as follows:

$$\nabla_{\theta^{\langle \mathcal{I}_j \rangle}} J(\theta^{\langle \mathcal{I}_j \rangle}) = \mathbb{E}_{\theta^{\langle \mathcal{I}_j \rangle}} \left[f(\pi_{\theta^{\langle \mathcal{I}_j \rangle}}) \nabla_\theta \log(p(\theta^{\langle \mathcal{I}_j \rangle})) \right]. \tag{3}$$

In the case of factored Gaussian distribution with the deviation σ, we can set $\theta^{\langle \mathcal{I}_j \rangle}$ as the mean vector. Then, the expected fitness of Eq. (2) is rewritten as follows:

$$\mathbb{E}_{\theta^{\langle \mathcal{I}_j \rangle}}[f(\pi_{\theta^{\langle \mathcal{I}_j \rangle}})] = \mathbb{E}_{\epsilon \sim \mathcal{N}^{\langle \mathcal{I}_j \rangle}(0,I)}[f(\pi_{\theta^{\langle \mathcal{I}_j \rangle} + \sigma\epsilon})]. \tag{4}$$

Practically, each individual, i.e., π_{ψ_i}, the actor in the population is produced by a perturbation operation $\psi_i = \theta^{\langle \mathcal{I}_j \rangle} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}^{\langle \mathcal{I}_j \rangle}(0, I)$. The average estimated gradient in this case according to Eq. (4) is shown as follows:

$$\nabla_{\theta^{\langle \mathcal{I}_j \rangle}} \mathbb{E}_{\theta^{\langle \mathcal{I}_j \rangle}}[f(\pi_{\theta^{\langle \mathcal{I}_j \rangle}})] \cong \frac{1}{\mu\sigma} \sum_{i=1}^{\mu} [f(\pi_{\theta^{\langle \mathcal{I}_j \rangle} + \sigma\epsilon_i}) \epsilon_i], \tag{5}$$

where μ is the population size. The estimated partial gradient enables searching for a good optimisation direction for each subproblem. Instead of independently optimising each subproblem, we introduce the concept of coevolution by the divide-and-conquer strategy, in which each subproblem is optimised iteratively, as shown in Figure 3. Using Figure 3 as an example, a neural network optimisation problem (weights) is decomposed into 3 subproblems (3 subsets), highlighted by red, green and blue, respectively. Each subproblem aims at optimising a subset of the weights. First, weights in red will be updated, while those in black will be frozen. Then, weights in green will be updated, while both weights in red and black will be frozen. Similar steps proceed until all weights are updated once. Optimising subproblems is not independent but in a cascade way. Optimisation of the current subproblem relies on the previous subproblem. The idea of coevolution builds the bridge between subproblems. The estimated gradient of the preceding subproblem can be regarded as momentum, which facilitates the subsequent optimisation. Additionally, the time complexity of the partially updating is $\mathcal{O}(|\theta|)$ with linearly scaling the number of parameters. This complexity is easy to get since each subproblem is the complimentary set of other subproblems. This complexity is much smaller than that of current gradient-based operators such as distilled crossover [2], which requires backpropagation with batches.

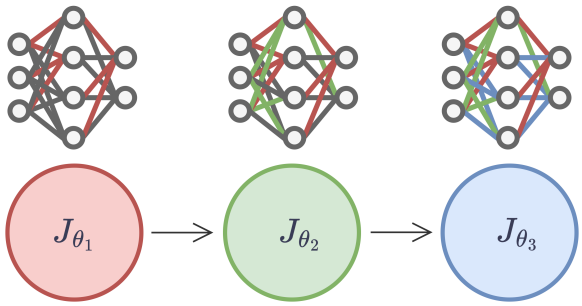

Figure 3. An example of partial updating via CC. Three subproblems are highlighted in red, green, and blue, respectively. For each subproblem, the policy inherits the partial gradient from the previous subproblem. Eventually, all parameters are updated once and only once.

3.3 *Leveraging temporal information*

To fully utilise the collected experiences, we choose soft actor-critic (SAC) [10] as the base algorithm in the RL loop, which optimises the policy in an off-policy way. Besides, the actor-critic architecture allows us to train the actor and critic separately. Hence, a population can be directly generated from an actor. The optimised actor after the CC loop is then used in policy improvement directly. SAC changes the objective of RL by adding the entropy term. The entropy of the policy is regarded as an extra rewarding signal. Larger entropy indicates a greater tendency for exploration. The Q function is parameterised by ϕ, and the target Q value using reparameterisation $\tilde{a}' \sim \pi_\theta(\cdot|s')$ is shown as follows:

$$y(r, s') = r + \gamma(\min_{j=1,2} Q_{\phi_j}(s', \tilde{a}') - \alpha_s \log \pi_\theta(\tilde{a}'|s')), \tag{6}$$

where α_s is the temperature coefficient. If s' is the end of the episode, then $y(r, s') = r$. The loss function of the Q-network is shown as follows:

$$L(\phi_i) = \mathop{\mathbb{E}}_{(s,a,r,s')} \left[(Q_{\phi_i}(s, a) - y(r, s'))^2 \right]. \tag{7}$$

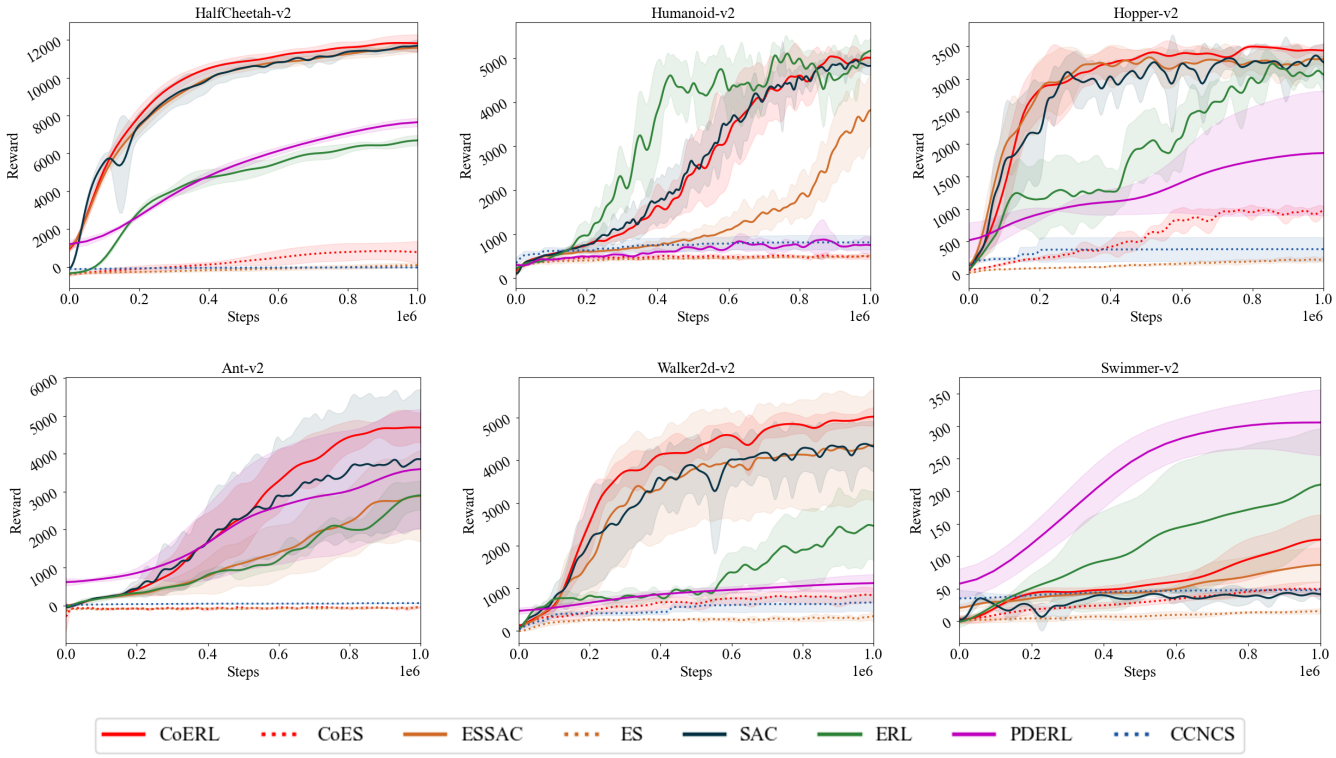

Figure 4. Training curves on six locomotion tasks. Each algorithm is trained for 1e6 timesteps with five different random seeds.

According to the additional entropy term, the objective of SAC is

$$\max_{\theta} \; J(\theta) = \mathbb{E}\left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta) - \alpha_s \log \pi_\theta(\tilde{a}_\theta|s)\right], \quad (8)$$

where $\tilde{a}_\theta$ is sampled via the reparameterisation trick.

4 Experiments

CoERL is compared with four advanced RL and ERL algorithms, namely SAC [10], ERL [11], PDERL [2], and CCNCS [39] on six benchmark locomotion tasks, including Ant-v2, HalfCheetah-v2, Humanoid-v2, Hopper-v2, Walker2d-v2, and Swimmer-v2 from Mujoco [31] integrated in the OpenAI gym [3]. Additionally, CoERL is split into three independent methods for conducting ablation study, namely cooperative coevolutionary evolution strategies (CoES), evolution strategies with soft actor-critic (ESSAC), and evolution strategies (ES) [26].

4.1 Settings

CoERL is implemented with the Tianshou framework [35]. The network structure is a fully connected neural network with a $\langle 256, 256 \rangle$ linear layer. γ is 0.99. In this paper, we apply random grouping [41, 39] and single best collaboration [24] for low cost and simplicity. The grouping number at each generation is randomly chosen from two, three and four, as Yang et al. [39] suggested. Hyperparameters of SAC, ERL, PDERL and CCNCS follow their original settings in [10, 11, 2, 39], respectively. Main hyperparameters are shown in Table 1. All algorithms are trained by 1e6 timesteps with five different random seeds. CoERL's hyperparameters are arbitrarily set, and remain the same values for all tasks.

4.2 Comparison results

Figure 4 shows the training curves assembled with five different random seeds. Our CoERL, highlighted with red colour, presents comparable performance and superior convergence on *HalfCheetah-v2*, *Hopper-v2*, *Ant-v2* and *Walker2d-v2*. Table 2 presents the final reward performance of CoERL and all comprised algorithms across six locomotion tasks. CoERL achieves the best average rewards on four tasks including *HalfCheetah-v2*, *Hopper-v2*, *Ant-v2* and *Walker2d-v2*. For example, in *Ant-v2*, CoERL obtains the best average reward of 5037.22, while the second-best algorithm SAC only gets 3654.16. Besides, CoERL achieves the highest average rank of 1.67, considering all six tasks. CoERL decomposes the neural network and searches for the partial gradient for each subproblem. Each parameter of the entire neural network is updated only once within an iteration, with a complexity of $\mathcal{O}(|\theta|)$. The partial gradient estimation does not rely on the delicate calculation of batches from the dataset but only requires the final outcome of evaluations, which can be easily sped up on CPUs according to Eq. (5).

Table 1. Hyperparameters for all compared algorithms. "-" denotes that the parameter is not involved.

Algorithm	Pop size μ	Learning rate Actor/Critic/Evolution
CoERL (ours)	6	1e-3/ 1e-3/ 1e-3
CoES	6	-/-/ 1e-3
ESSAC	6	1e-3/ 1e-3/ 1e-3
ES (Salimans et al. [26])	6	-/-/ 1e-3
SAC (Haarnoja et al. [10])	-	1e-3/ 1e-3/-
ERL (Khadka and Tumer [11])	10	3e-4/ 1e-4/-
PDERL (Bodnar et al. [2])	10	5e-4/ 5e-3/-
CCNCS (Yang et al. [39])	6	-/-/-

Table 2. Final reward performances with five different seeds in six locomotion environments. The bold number denotes the highest average value of each column. "Avg. Rank" denotes the average rank across all six tasks. Lower rank indicates better overall performance. CoERL shows the best performance with the highest average rank 1.67.

Algorithm	HalfCheetah-v2 Avg. ± Std	Humanoid-v2 Avg. ± Std	Hopper-v2 Avg. ± Std	Avg. Rank
CoERL (ours)	**11959.63 ± 250.15**	4642.05 ± 762.38	**3414.45 ± 100.32**	**1.67**
CoES	942.63 ± 419.89	482.47 ± 78.36	991.12 ± 25.77	6.33
ESSAC	11597.69 ± 209.71	3931.32 ± 1309.69	3176.29 ± 481.59	3.50
ES (Salimans et al. [26])	100.32 ± 135.40	490.86 ± 34.59	221.77 ± 41.11	7.67
SAC (Haarnoja et al. [10])	11774.61 ± 255.86	**4887.86 ± 296.71**	2842.68 ± 528.56	3.17
ERL (Khadka and Tumer [11])	6790.15 ± 582.91	4677.19 ± 1054.07	2998.35 ± 384.52	3.33
PDERL (Bodnar et al. [2])	7845.97 ± 285.62	765.55 ± 195.15	1886.00 ± 988.68	3.83
CCNCS (Yang et al. [39])	-27.73 ± 28.11	813.44 ± 161.77	384.13 ± 182.82	6.50

Algorithm	Ant-v2 Avg. ± Std	Walker2d-v2 Avg. ± Std	Swimmer-v2 Avg. ± Std	Avg. Rank
CoERL (ours)	**5037.22 ± 192.01**	**4962.80 ± 412.39**	128.90 ± 40.13	**1.67**
CoES	-26.53 ± 7.40	872.44 ± 116.53	55.89 ± 11.42	6.33
ESSAC	2927.98 ± 1496.94	4228.00 ± 1283.25	94.87 ± 25.12	3.50
ES (Salimans et al. [26])	-71.04 ± 13.72	370.08 ± 134.90	15.91 ± 7.77	7.67
SAC (Haarnoja et al. [10])	3654.16 ± 1767.25	4397.42 ± 506.47	37.95 ± 15.59	3.17
ERL (Khadka and Tumer [11])	2982.24 ± 438.70	2790.96 ± 955.50	214.50 ± 85.49	3.33
PDERL (Bodnar et al. [2])	3730.17 ± 1484.37	1183.03 ± 247.26	**305.35 ± 58.09**	3.83
CCNCS (Yang et al. [39])	62.29 ± 10.66	669.85 ± 204.35	47.71 ± 3.01	6.50

We also notice that CoERL does not achieve outstanding performance in *Swimmer-v2*, although its performance has been improved from 55.89 to 94.87 and 128.90 with the help of CC and the utility of temporal information by ESSAC. The poor performance of SAC might explain this case, as it only achieves 37.95, surpassing only the 15.91 achieved by ES. Given that CoERL is built upon SAC, it is not surprising that CoERL encounters the same local optimum as SAC, despite ultimately reaching 128.90. PDERL, the improved version of ERL only get 765.55 in *Humanoid-v2*, while our CoERL gets an average reward of 4642.05 and ERL gets 4677.19. It is attributed to the high dimensions of *Humanoid-v2* for the peculiar performance of PDERL. The observation dimension of *Humanoid-v2* is 376, which is the largest among six tasks. The distilled crossover maintains the consistency of behaviour spaces while exchanging parameters using supervised learning techniques. However, this improvement leads to higher computational demands during backpropagation. Our CoERL, in contrast, searches for the partial gradient at a lower computational cost than backpropagation, while still preserving consistency.

Moreover, none of the pure evolution-based algorithms, such as ES [26] and CCNCS [39], demonstrates overall promising performance, even though CoES and CCNCS achieve superior average rewards than SAC in *Swimmer-v2* with 55.89 and 47.71, respectively. As discussed previously, evolution-based algorithms have long struggled with the scalability issue [18]. And the temporal experiences are barely utilised in evolution, resulting in inefficient training. Therefore, more computational time is required to converge. Our CoERL overcomes those issues and achieves the highest rank with 1.67.

4.3 Ablation study

To fully verify the effectiveness of CoERL, we examine the contribution of its components independently. The evolution part is split into coevolution strategies (CoES) and evolution strategies (ES). Regarding the RL part, we consider pure RL algorithm, SAC, and the simplified version without coevolution, ESSAC. The performance of CoERL and its ingredients, including CoES, ES, SAC, and ESSAC, are shown in Table 2 and Figure 4. It is evident from the experimental results that none of the ablated algorithms like ES, CoES and ESSAC outperforms CoERL, indicating their unique contributions to CoERL.

Evolution-based algorithms such as CoES and ES only utilise the outcome of the evaluation as the fitness. Although EAs show some advantages in tackling the credit assignment problem with sparse reward [26], it wastes adequate temporal information collected during evaluation. On the other hand, modern MDP-based RL algorithms tend to use temporal information, i.e., transition-based experiences, while ignoring the long-term cumulative reward. This dilemma is similar to the trade-off between Monte-Carlo sampling and temporal difference [30], which seeks to balance between bias and variance of the estimated value function. In our case, the usage of temporal information becomes an issue as traditional evolution-based algorithms merely provide a solution to utilise it.

Instead of a single evolution loop, CoERL maintains an additional MDP-based RL loop for better use of temporal information. The RL loop reuses the experiences collected by the actors in the evolution. Since the evolution loop still inherits the surviving technique based on fitness, a variation between the target policy and the behaviour policy, i.e., individuals, is introduced. So the choice of MDP-based RL has to be the off-policy version [11].

Intuitively, the additional RL loop proceeds to optimise the policy, which achieves an efficient sample utility for picking up the wasted experiences. The policy is improved using value approximation in a fine-grained way. At the same time, when looking into the way of collecting experiences, we find that the underlying mechanism of the reproduction shows an advantage on exploration, similar to noisy exploration [5]. New individuals are generated through proximal perturbation in CoERL, ensuring a diverse range of experiences.

4.4 Inheriting behaviour space

To further analyse the mechanism of CoERL, we visualise the behaviour space using t-SNE [32] during training. The t-SNE compresses the high-dimensional observation space into a two-dimensional space. *Swimmer-v2* is chosen for a case study. Figure 5 shows an example of the visualisation in the case of being decomposed into four subproblems. From left to right, the first figure is the behaviour feature map of the agent after optimising weight subset 1, then the second one shows after optimising weight subset 2, and so

Table 3. Average running time (minutes) of algorithms over five trials with different random seeds,

Algorithm	HalfCheetah-v2	Humanoid-v2	Hopper-v2	Ant-v2	Walker2d-v2	Swimmer-v2
CoERL (ours)	302.44	237.63	219.72	238.66	313.34	54.43
CoES	10.11	19.51	15.96	21.51	15.24	19.48
ESSAC	298.89	234.75	239.38	224.98	302.8	53.18
ES (Salimans et al. [26])	7.31	15.33	14.32	14.93	12.48	16.12
SAC (Haarnoja et al. [10])	302.69	263.17	228.62	241.13	316.09	52.95
ERL (Khadka and Tumer [11])	1105.44	1309.86	1167.94	1102.4	1079.94	1028.3
PDERL (Bodnar et al. [2])	222.36	626.37	233.32	245.67	207.37	225.93
CCNCS (Yang et al. [39])	6.02	46.33	41.87	29.73	34.97	5.8

on. Each subproblem does one full step before the next subproblem in CoERL, however, this update is only limited to the subproblem (denoted by parameter indices $\langle \mathcal{I}_j \rangle$). The gradient of the preceding subproblem remains an implicit effort on the optimisation direction of the subsequent subproblem, acting as momentum. This effort is easily observed in Figure 5, where the first feature map shares a high similarity with the third one (from left to right), as well as the second and fourth ones. This phenomenon implies that the behaviour space is inherited during optimisation.

Benefiting from the cooperative coevolution strategy and partial gradients, the offspring policy inherits the behaviour space of its parents after being updated in the last subproblem. Since CoERL only optimises the parameters of the subproblem each time, the remaining quotient set of parameters acts as a "memory buffer". After the updates, some old neural activations of the network in the memory buffer still connect, resulting in the emergence of certain behaviours of the new policy. Additionally, the partial gradient can be considered as a form of proximal variation [27]. Instead of aimlessly perturbing the policy, CoERL searches for the partial gradient within a proximal area, which ensures policy updates within a promising range. Then, the sampled individuals provide a certain optimisation direction, which is assembled as a vector according to Eq. (5).

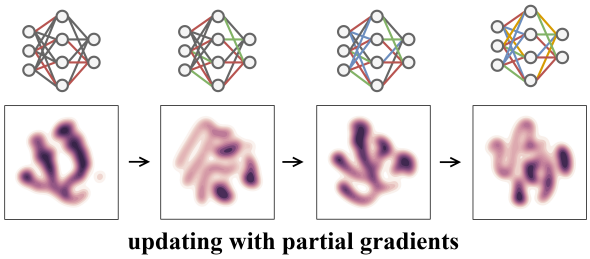

Figure 5. Four exclusive subproblems are highlighted in red, green, blue and yellow, respectively. The feature maps present the behaviour spaces, reduced by t-SNE [32], of the policy after optimised in subproblems.

4.5 Direct coordination or indirect coordination

A possible cause for the failure of CoERL in *Swimmer-v2* could be attributed to the coordination between the CC loop and the RL loop. In ERL, the learning agent injects gradient information by replacing the worst individual in the population [11]. This indirect coordination requires an extra roll-out by the learning agent itself and introduces a trade-off when choosing individuals. Our CoERL avoids these issues by sampling the population at each generation instead of maintaining a permanent population, allowing for more direct coordination. The optimised policy can directly access the evolution loop. However, we have to admit that this direct coordination may stuck in a local optimum if the base learning algorithm is not a good partner, even though CoERL has demonstrated outstanding performances on almost all tasks. Balancing the trade-off between direct and indirect coordination is worth considering as a future research direction.

4.6 Runtime analysis

Table 3 presents the average running times of algorithms in five trials. The results show that the running time of CoERL is close to SAC and much shorter than ERL. CoERL is particularly designed for addressing the scalability issue through random grouping, which actually does not introduce additional computational complexity via random grouping. The complexity of the proposed partial updating is $\mathcal{O}(|\theta|)$, which should not be greater than the one-time policy gradient via backpropagation. CCNCS has the shortest running time as it doesn't apply gradient techniques and only updates weights a few times in each generation. However, its performance is relatively poor.

5 Conclusion

In this paper, we propose CoERL, a novel cooperative coevolutionary reinforcement learning algorithm to address the scalability issues and enhance the efficiency during training. CoERL decomposes the policy optimisation problem into multiple subproblems using a cooperative coevolutionary strategy. For each subproblem, CoERL searches for partial gradients to update the policy. This decomposition, coupled with the use of partial gradients, ensures consistency between the behaviour spaces of parents and offspring at a reduced cost. In contrast to traditional evolution-based approaches that discard experiences, CoERL capitalises on the collected experiences within the population, as a novel hierarchy based on cooperative coevolution. Extensive experiments on six locomotion tasks demonstrate that CoERL outperforms seven state-of-the-art algorithms and baselines. The contributions of CoERL's components are also verified through an ablation study. In the future, CoERL can be extended using knowledge-based grouping techniques and combined with other ERL algorithms like [44] to cope with the exploration-exploitation dilemma. Furthermore, exploring the explainable decomposition in neural networks via visualisation and quantification could be another interesting direction. It is also worth investigating a thorough theoretical analysis and hyperparameter sensitivity analysis.

References

[1] H. Bai, R. Cheng, and Y. Jin. Evolutionary reinforcement learning: A survey. *Intelligent Computing*, 2:0025, 2023.

[2] C. Bodnar, B. Day, and P. Lió. Proximal distilled evolutionary reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 3283–3290, 2020.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint:1606.01540*, 2016.

[4] M.-L. Cauwet, J. Liu, B. Rozière, and O. Teytaud. Algorithm portfolios for noisy optimization. *Annals of Mathematics and Artificial Intelligence*, 76:143–172, 2016.

[5] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, pages 1–21, 2018. URL https://openreview.net/forum?id=rywHCPkAW.

[6] E. Galván and P. Mooney. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Transactions on Artificial Intelligence*, 2(6):476–493, 2021.

[7] T. Gangwani and J. Peng. Genetic policy optimization. In *International Conference on Learning Representations*, pages 1–16, 2018. URL https://openreview.net/forum?id=ByOnmlWC-.

[8] E. Glorieux, B. Svensson, F. Danielsson, and B. Lennartson. Improved constructive cooperative coevolutionary differential evolution for large-scale optimisation. In *IEEE Symposium Series on Computational Intelligence*, pages 1703–1710. IEEE, 2015.

[9] M. Gong, J. Liu, A. K. Qin, K. Zhao, and K. C. Tan. Evolving deep neural networks via cooperative coevolution with backpropagation. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):420–434, 2020.

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

[11] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *International Conference on Neural Information Processing Systems*, page 1196–1208. Curran Associates Inc., 2018.

[12] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, and K. Tumer. Collaborative evolutionary reinforcement learning. In *International Conference on Machine Learning*, pages 3341–3350. PMLR, 2019.

[13] J. Lehman, J. Chen, J. Clune, and K. O. Stanley. ES is more than just a traditional finite-difference approximator. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 450–457, 2018.

[14] J. Lehman, J. Chen, J. Clune, and K. O. Stanley. Safe mutations for deep and recurrent neural networks through output gradients. In *Genetic and Evolutionary Computation Conference*, pages 117–124, 2018.

[15] F.-Y. Liu, Z.-N. Li, and C. Qian. Self-guided evolution strategies with historical estimated gradients. In *International Joint Conferences on Artificial Intelligence*, pages 1474–1480, 2020.

[16] J. Liu and K. Tang. Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 350–357. Springer, 2013.

[17] J. Liu and O. Teytaud. Meta online learning: experiments on a unit commitment problem. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 485–490, 2014.

[18] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, and Z. Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2018.

[19] E. Marchesini, D. Corsi, and A. Farinelli. Genetic soft updates for policy evolution in deep reinforcement learning. In *International Conference on Learning Representations*, pages 1–15, 2020.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[21] D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[22] Y. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017.

[23] L. Panait. Theoretical convergence guarantees for cooperative coevolutionary algorithms. *Evolutionary Computation*, 18(4):581–615, 2010.

[24] M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Evolutionary Computation*, pages 249–257. Springer, 1994.

[25] Pourchot and Sigaud. CEM-RL: Combining evolutionary and gradient-based methods for policy search. In *International Conference on Learning Representations*, pages 1–13, 2019. URL https://openreview.net/forum?id=BkeU5j0ctQ.

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

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

[28] O. Sigaud. Combining evolution and deep reinforcement learning for policy search: A survey. *ACM Transactions on Evolutionary Learning*, 3(3):1–20, 2023.

[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550: 354–359, Oct. 2017.

[30] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

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

[32] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

[33] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[34] Y. Wang, T. Zhang, Y. Chang, X. Wang, B. Liang, and B. Yuan. A surrogate-assisted controller for expensive evolutionary reinforcement learning. *Information Sciences*, 616(C):539–557, nov 2022. ISSN 0020-0255.

[35] J. Weng, H. Chen, D. Yan, K. You, A. Duburcq, M. Zhang, Y. Su, H. Su, and J. Zhu. Tianshou: A highly modularized deep reinforcement learning library. *The Journal of Machine Learning Research*, 23(1):12275–12280, 2022.

[36] S. Whiteson. Evolutionary computation for reinforcement learning. *Reinforcement Learning: State-of-the-art*, pages 325–355, 2012.

[37] Y. Wu, X. Peng, H. Wang, Y. Jin, and D. Xu. Cooperative coevolutionary CMA-ES with landscape-aware grouping in noisy environments. *IEEE Transactions on Evolutionary Computation*, 27(3):686–700, 2023. doi: 10.1109/TEVC.2022.3180224.

[38] P. Yang, K. Tang, and X. Yao. Turning high-dimensional optimization into computationally expensive optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):143–156, 2017.

[39] P. Yang, H. Zhang, Y. Yu, M. Li, and K. Tang. Evolutionary reinforcement learning via cooperative coevolutionary negatively correlated search. *Swarm and Evolutionary Computation*, 68:100974, 2022.

[40] P. Yang, L. Zhang, H. Liu, and G. Li. Reducing idleness in financial cloud services via multi-objective evolutionary reinforcement learning based load balancer. *SCIENCE CHINA Information Sciences*, pages 1–20, 2023. doi: https://doi.org/10.1007/s11432-023-3895-3.

[41] Z. Yang, K. Tang, and X. Yao. Multilevel cooperative coevolution for large scale optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1663–1670. IEEE, 2008.

[42] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 8(4):539–567, 1993.

[43] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[44] Q. Zhu, X. Wu, Q. Lin, and W.-N. Chen. Two-stage evolutionary reinforcement learning for enhancing exploration and exploitation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20892–20900, 2024.