Strategic Evolutionary Reinforcement Learning with Operator Selection and Experience Filter

Kaitong Zheng, *Student Member, IEEE,* Ya-Hui Jia, *Member, IEEE,* Kejiang Ye, *Senior Member, IEEE*
Wei-Neng Chen, *Senior Member, IEEE*

***Abstract*—The shared replay buffer is the core of synergy in evolutionary reinforcement learning (ERL). Existing methods overlooked the objective conflict between population evolution in evolutionary algorithm and ERL, leading to poor quality of the replay buffer. In this paper, we propose a strategic evolutionary reinforcement learning algorithm with operator selection and experience filter to address the objective conflict issue and improve the synergy from three aspects. 1) An operator selection strategy is proposed to enhance the performance of all individuals, thereby fundamentally improving the quality of experiences generated by the population. 2) An experience filter is introduced to filter the experiences obtained from the population, maintaining the long-term high quality of the buffer. 3) A dynamic mixed sampling strategy is introduced to improve the efficiency of RL agent learning from the buffer. Experiments in four MuJoCo locomotion environments and three Ant-Maze environments with deceptive rewards demonstrate the superiority of the proposed method. Additionally, the practical significance of the proposed method is verified on a low-carbon multi-energy microgrid energy management task.**

***Index Terms*—Evolutionary reinforcement learning, evolutionary algorithms, deep reinforcement learning, replay buffer.**

I. Introduction

DEEP Reinforcement Learning (DRL) algorithms have achieved significant success in numerous fields, such as games [1], [2], robotic systems [3], [4], and learning-based control [5], [6]. The effectiveness of reinforcement learning highly depends on the hand-crafted design of the reward functions [7], [8]. In many real-world scenarios, designing a reward function that provides timely and accurate feedback is challenging [9]. For example, in resource scheduling tasks, rewards are typically sparse and delayed, as they are only received after a sequence of operations is completed [10]. When rewards are sparse and delayed, the learning efficiency of DRL decreases drastically due to poor exploration capability [11], [12]. Evolution Algorithms (EAs), a class of gradient-free optimization algorithms [13] including genetic algorithm [14] and evolutionary strategy [15], have recently emerged as a promising alternative to DRL [16], [17]. Due to the population-based and gradient-free random search characteristics, EAs are indifferent to the sparsity of reward and are robust to scenarios with long-time horizons [18], [19]. Meanwhile, EAs have the advantages of maintaining a beneficial exploration and improving robustness, contributing to a more stable convergence [20]. However, the gradient-free EAs encounter challenges such as high sample complexity and slow convergence rate when tackling high-dimensional problems [21], [22]. DRL and EAs have complementary strengths, and their combination has emerged as a promising research direction.

Leveraging the strengths of both DRL and EA synergistically, Khadka *et al.* [18] proposed a new RL paradigm called Evolutionary Reinforcement Learning (ERL). It maintains an actor-network population evolved through multi-point crossover and Gaussian mutation, and an RL agent trained using gradient-based optimization. The population and RL agent are bridged via a shared replay buffer and a synchronization mechanism, effectively accelerating the learning process. Subsequently, many ERL algorithms have been proposed to pursue better efficacy. Some ERL algorithms prioritize increasing the efficiency of population evolution, in which several advanced crossover and mutation operators are proposed, such as proximal mutation [23], distillation crossover [23], and distillation mutation [24]. Cross Entropy Method (CEM) is also used as a method for evolving populations [25], [26]. Some ERL algorithms improve performance by modifying the integration of population and RL agent. ERL-Re^2 [27] enables the actor networks in the population to share nonlinear state representation layers with the RL agent while retaining a separate linear policy layer. In CoERL [28], rather than maintaining a population of actor networks, cooperative coevolution is used to evolve the actor network of the RL agent. ERL-TD [24] leverages multiple critic networks and a truncated variance strategy to mitigate overestimation bias and improve the learning efficiency of the RL agent.

Although existing ERL algorithms vary in many aspects, most of them still follow the basic experience generation and utilization method of the original ERL framework that the experiences generated by a gradient-free population are indiscriminately injected into a shared replay buffer, while a gradient-based RL agent learns from the buffer using a uniform random sampling strategy. However, this method has

This work is supported in part by the National Natural Science Foundation of China under Grants 62206098, the Guangdong Basic and Applied Basic Research Foundation 2023A1515012896, and the Guangzhou Basic and Applied Basic Research Foundation 2023A04J1692. (*Corresponding author: Ya-Hui Jia; Kejiang Ye.*)

Kaitong Zheng is with School of Future Technology, South China University of Technology, Guangzhou, China, and also with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. (email: 202210191867@mail.scut.edu.cn)

Ya-Hui Jia is with School of Future Technology, South China University of Technology, Guangzhou, China. (email: jia.yahui@foxmail.com)

Kejiang Ye is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. (email: kj.ye@siat.ac.cn)

Wei-Neng Chen is with School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. (email: cwnraul634@aliyun.com)

overlooked the quality of the experience generated by the population. Existing ERL algorithms tend to focus on the good-performing individuals in the population while paying less attention to the detrimental effects that poorly performing individuals may introduce. Due to the stochastic nature of EAs, it is inevitable that many poorly performing individuals will emerge during the evolution process. In the basic experience utilization method, the low-quality experiences they generate are also injected into the shared replay buffer. An excessive accumulation of low-quality experiences can significantly degrade the quality of the buffer. Meanwhile, there is usually an experience distribution mismatch between the experiences generated by the population and the RL agent [30]. The traditional uniform random sampling strategy used in the existing ERL algorithms may fail to fully take advantage of the shared replay buffer. Hence, the learning efficiency of the RL agent is hindered, thereby affecting the overall synergy.

To address the aforementioned issues, we propose a Strategic Evolutionary Reinforcement Learning algorithm with Operator Selection and Experience Filter (SERL-OS-EF), which aims to improve the efficiency of information flow between the population and the RL agent. In contrast to existing methods, our method emphasizes the performance of all individuals during the evolutionary process, rather than solely focusing on the good-performing individuals. Additionally, we place significant emphasis on maintaining the long-term quality of the shared replay buffer from experience generation and injection. Meanwhile, we design a dynamic mixed sampling strategy to replace the conventional uniform random sampling method, mitigating the experience distribution mismatch problem and maximizing the utilization of the shared replay buffer.

The main contributions of this work are summarized as follows:

- We investigate the issue of the bad influence of accumulating a large number of low-quality experiences in the shared replay buffer and attribute this issue to the objective conflict between population evolution in EA and ERL.
- To enhance the quality of the shared replay buffer, an operator selection strategy is proposed to enhance the overall quality of the population, thereby generating more high-quality experiences. Meanwhile, an experience filter is proposed to filter out low-quality experiences from the population, which is crucial for maintaining the long-term high quality of the buffer.
- A dynamic mixed sampling strategy is employed to promote the learning efficiency of the RL agent, thereby strengthening the synergy between the EA and RL components within the ERL framework.
- We compare our method with the state-of-the-art methods in four MuJoCo environments, three Ant-Maze environments, and a practical low-carbon multi-energy microgrid energy management task. The results show that our method outperforms existing methods, achieving state-of-the-art results in the Ant-Maze environments and the practical task, while also achieving the best average rank in four MuJoCo environments.

The rest of this paper is organized as follows. Section II introduces some critical techniques related to ERL and existing studies of ERL. Section III analyzes the objective conflict between population evolution in EA and ERL, and demonstrates the proposed method in detail. Section IV presents the empirical studies on benchmark and real-world problems. Finally, the conclusion is drawn in Section V.

II. Background

This section introduces the notations and fundamental concepts of ERL and related work.

A. Evolutionary Reinforcement Learning

In ERL, as shown in Fig. 1, a population of actor networks and an actor-critic RL agent RL_{agent}, consisting of a RL_{actor} and a RL_{critic}, are initialized. The actor population and RL_{agent} adopt different mechanisms for optimization. The population evolution follows the framework of EAs. The return obtained from the interaction between each individual and the environment is considered as the fitness value of the individual. Some individuals are selected based on their fitness values to generate new individuals. To ensure the quality of offspring, individuals with higher fitness values have a greater probability of being selected. EA operators directly manipulate the selected individuals, i.e. evolving the neural networks they represent. In the evolution process, a diverse set of experiences is generated and saved into a fixed-capacity replay buffer, which is shared by the population and RL_{agent}. RL_{agent} is trained by a gradient-based optimizer through sampling experiences from the replay buffer.

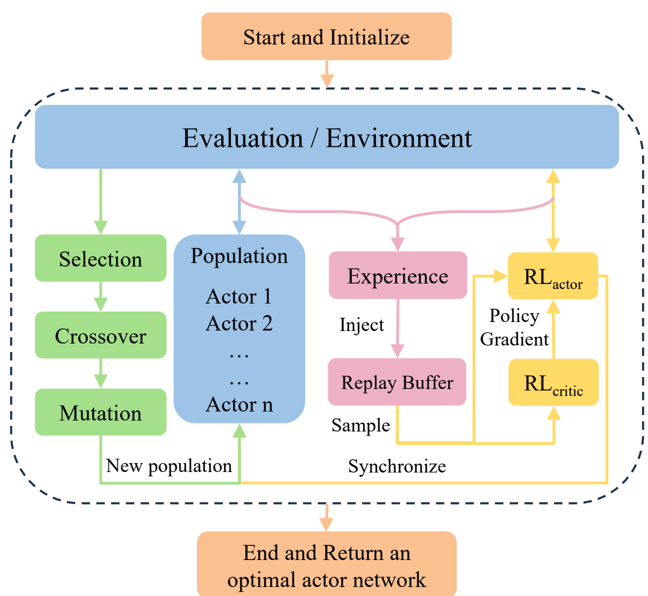


Fig. 1. Framework of ERL.

The shared replay buffer and synchronization are the key components of the bidirectional transfer of information in the ERL framework. The shared replay buffer facilitates the flow of information from the population to RL_{agent} . Synchronization facilitates the flow of information from RL_{agent} to the

TABLE I
A COMPARISON OF SOME REPRESENTATIVE ERL ALGORITHMS.

Algorithm	Operators	RL type	Experience injection	Sampling methods
ERL [18]	multi-point crossover, Gaussian mutation	DDPG	indiscriminate	uniform random sampling
PDERL [23]	distillation crossover, proximal mutation	DDPG	indiscriminate	uniform random sampling
CEM-RL [25]	CEM	TD3	indiscriminate	uniform random sampling
NERL [26]	CEM, novelty search	TD3	indiscriminate	uniform random sampling
SUPE-RL [29]	Gaussian mutation	Rainbow/PPO	indiscriminate	uniform random sampling
ERL-Re^2 [27]	multi-point crossover, Gaussian mutation	TD3 with shared network	indiscriminate	uniform random sampling
ERL-TD [24]	distillation mutation	SAC with truncated variance	indiscriminate	uniform random sampling
CoERL [28]	cooperative coevolution	SAC	indiscriminate	uniform random sampling
TR-ERL [30]	canonical evolution strategy	TD3	indiscriminate	fix mixed sampling
Ours	opposite-based proximal mutation gradient optimization operator	TD3	discriminative filtering	dynamic mixed sampling

population. The gradient-based RL_{agent} have a higher sample efficiency [18]. Synchronization will offer a more promising parameter space for the population to explore. Specifically, synchronization refers to replacing the worst individual in the population with the actor network of RL_{agent} at regular intervals.

B. Related work

Most ERL algorithms typically involve an evolutionary loop alongside a reinforcement learning loop [31]. In the RL loop, RL_{agent} usually updates with gradient methods, where the specific update strategy varies across different actor-critic algorithms. In the evolutionary loop, the population is used to generate a diverse set of experiences and evolved by gradient-free methods. Population evolution methods are generally classified into two main categories: genetic algorithm and evolution strategy.

In the original ERL, multi-point crossovers and Gaussian mutation are used to generate a new population. Q-filtered distillation crossovers and proximal mutation based on back-propagation are proposed to mitigate catastrophic forgetting in multi-point crossovers and Gaussian mutation in PDERL [23]. To improve the performance of mutated offspring, distilled mutation [24] utilizes elite individuals in the population to provide directional guidance for the mutation process through policy distillation. In Genetic-Gated Networks (G2N) [32] and Soft Updates for Policy Evolution (SUPE-RL) [29], the new population is obtained by conducting crossover and mutation on RL_{agent}. To address the scalability issue of ERL, CoERL [28] decomposes the policy optimization problem into multiple subproblems and optimizes them sequentially. During this process, it adaptively adjusts the perturbation magnitude of subproblems, which can be regarded as an approximate mutation strategy. Besides genetic algorithms, evolutionary strategies are also widely applied in ERL algorithms [30], [33]. CEM [34] is an evolution strategy algorithm that optimizes the parameters by solving a sequence of auxiliary smooth optimization problems using Kullback-Leibler cross-entropy. Pourchot *et al.* proposed a CEM-RL algorithm [25], which combines TD3 and CEM. To avoid premature convergence, a novelty search is introduced into the CEM-RL framework to encourage individuals to search an entirely new policy space [26]. Other follow-ups of CEM-RL are CEM-SAC [35], which is a hybridization between CEM and Soft-Actor-Critic (SAC) [36], and CEM-ACER [37], which combines CEM and Actor-Critic with Experience Replay (ACER) [38].

Table I summarizes the characteristics of some representative algorithms. It is evident that the differences among existing ERL algorithms primarily lie in the EA operators and RL types, while they remain identical in the experience injection method and sampling method. This phenomenon stems primarily from existing ERL algorithms that directly incorporate EAs into the ERL framework, overlooking the objective conflict between population evolution in EA and ERL. In the next section, we will reveal the objective conflict and its detrimental impact on the shared replay buffer and the synergy in ERL. We address these issues by proposing a new population evolution method, improving the quality of the shared replay buffer, and implementing a more effective experience sampling method.

III. PROPOSED METHOD

This section begins with an analysis of the objective conflict between population evolution in EA and ERL. Then, we present the overall framework of our method and provide a comprehensive explanation, followed by a thorough description of the specific implementation details. For better understanding and reference, we have summarized the notations used in our method and presented them in Table II.

A. Objective conflict between population evolution in EA and ERL

We investigate the existence of the objective conflict from both theoretical and empirical perspectives. First, the objectives of population evolution in EA and ERL are defined as follows:

$$\theta^* = \arg \max_{\theta \in \bigcup_{t=1}^{T} \mathcal{P}^{(t)}} F(\theta) \tag{1}$$

$$\underset{\mathcal{P}^{(t)}}{\text{maximize}} \left[F(\mathcal{P}^{(t)}), \mathcal{D}(\mathcal{P}^{(t)}) \right], \quad t = 1, 2, \ldots, T. \tag{2}$$

where t and T are the current and maximum number of iteration, respectively. θ denotes an individual within the population across generations. θ^* is the best-performing individual. $\mathcal{P}^{(t)}$ denotes the population in t-th iteration. $F(\cdot)$ and $\mathcal{D}(\cdot)$ are the fitness and diverse functions, respectively.

TABLE II
NOTATION LIST

Notation	Definition
RL_{agent}	Gradient-based reinforcement learner in ERL
RL_{actor}	The actor-network in the RL_{agent}
RL_{critic}	The critic network in the RL_{agent}
t, $t-1$, $t+1$	The current, previous, and next iterations
T	The maximum number of iteration
N	The size of the population
ψ	The elite fraction of the population
δ	The threshold for identifying low-quality individuals
P_e	The performance enhancement probability
P_d	The performance degradation probability
P_s	The performance stagnation probability
ζ, p_0	The parameter and initial probability for P_e
γ	The degradation-induced low-quality probability
γ, η	The stagnation-induced low-quality probability
ρ	The proportion of low-quality individuals
RB_{pop}	The replay buffer of the population
RB_l	The replay buffer for low-quality individuals
RB_h	The replay buffer for high-quality individuals
$\mathcal{B}$, $\mathcal{B}_l$, $\mathcal{B}_h$	The mini-batch from RB_{pop}, RB_l, RB_h
ϕ	The parameter of critic network
Sub_i	The i-th subpopulation
S_t^i, S_{t+1}^i	the states of Sub_i in t-th and $t+1$-th iterations
$S_t^{sub_i}$	The information of Sub_i in t-th iteration
S_t^{pop}	The information of population in t-th iteration
FB_t^i	The best fitness value of Sub_i in t-th iteration
FA_t^i	The average fitness of Sub_i in t-th iteration
FBD_t^i	FB_t^i - FB_{t-1}^i
FAD_t^i	FA_t^i - FA_{t-1}^i
FB_t^{pop}	The best fitness values of the population in t-th iteration
FA_t^{pop}	The average fitness values of the population in t-th iteration
$Norm()$, abs	Normalization function and absolute value function
a_t^i	The action for Sub_i in current iteration
R	The reward obtained by taking action a_t^i in state S_t^i
s_e	The sensitivity of the output to the weights.
$\bigtriangledown$	Gradient operator
A	The dimension of the output action
N_M	The batch size in opposite-based proximal mutation
μ	The actor-network in the mutation operator
θ	The parameters of actor-network
σ	The mutation magnitude parameter
α	The scaling factor in mutation
F_a, F_b	The fitness values before and after the basic mutation
F_i	The fitness value of individual i
BF	The best fitness of the population
β	The parameter controlling filter intensity
$d_{D_{rl}}(s,a)$	The batches sampled from the RL_{agent} buffer
$d_{D_{pop}}(s,a)$	The batches sampled from the population buffer
$\hat{d}(s,a)$	The final batch used by the RL_{agent}
m	The proportion sampled from the RL_{agent} buffer
k	The parameter controlling the range of m

It is evident that the objective of the EA is to find the best-performing individual. Hence, not all individuals in the population need to perform well in EAs, as ultimately only the best individual is chosen to serve as the final solution. However, the objective of the ERL is to generate more diverse and beneficial experiences throughout the evolutionary process, which focus on the overall performance of the entire population. The root of the objective conflict in EA and ERL lies in the accumulation effect of low-quality individuals during population evolution. In EA, low-quality individuals are continuously eliminated through the evolutionary process, and once discarded, their influence on the algorithm effectively ceases. However, in ERL, low-quality individuals interact with the environment and contribute experiences to the shared replay buffer. These experiences may persist over time and continue to influence policy updates, even after the individuals themselves have been discarded. It is worth noting that due to the stochastic nature of EAs, new low-quality individuals are continuously generated even as old ones are discarded. Hence, the cumulative negative impact caused by low-quality individuals should not be underestimated.

Next, we employ mathematical analysis to investigate the accumulation of low-quality individuals in EA. In each iteration, the population consists of N individuals. The proportion of elite individuals inherited from the previous generation within the current population is denoted as ψ and the remaining $(1-\psi)N$ individuals generated by crossover and mutation. We define a low-quality individual as one whose fitness falls below a threshold (e.g., $\delta = 0.1$) of the current best fitness. As the population converges, the probability (P_e) of further enhancement for individuals gradually decreases. Meanwhile, we assume that the probability (P_d) of degradation remains constant and the probability (P_s) of stagnation increases.

$$P_e(t) = p_0 e^{-\zeta t}, \zeta > 0 \tag{3}$$

$$P_d(t) = p_d \tag{4}$$

$$P_s(t) = 1 - e^{-\zeta t} - p_d \tag{5}$$

where p_0 denotes the initial probability of P_e and p_d is a constant.

We define $q(t)$ as the proportion of low-quality individuals in t-th iteration, mainly derived from individuals with degraded or stagnant performance. Thus, it can be approximated by:

$$q(t) \approx \gamma \cdot P_{\mathrm{d}}(t) + \eta \cdot P_{\mathrm{s}}(t) \tag{6}$$

where $\gamma, \eta \in [0,1]$ denote the probabilities of falling into the low-quality region under performance degradation and stagnation, respectively. The expected number of low-quality individuals in t-th iteration is given by:

$$\mathbb{E}[L^{(t)}] = (1-\psi)N\left(\gamma p_d + \eta(1 - p_0 e^{-\zeta t} - p_d)\right) \tag{7}$$

Consequently, the cumulative number of low-quality individuals up to generation T becomes:

$$C^{(T)} = \sum_{t=1}^{T} \mathbb{E}[L^{(t)}] \tag{8}$$

$$= (1-\psi)N \sum_{t=1}^{T} \left(\gamma p_d + \eta(1 - p_0 e^{-\zeta t} - p_d)\right) \tag{9}$$

$$= (1-\psi)N \sum_{t=1}^{T} \left((\gamma-\eta)p_d + \eta(1 - p_0 e^{-\zeta t})\right) \tag{10}$$

$$= (1-\psi)N \left[T((\gamma-\eta)p_d + \eta) - \eta p_0 \cdot \frac{e^{-\zeta}(1-e^{-\zeta T})}{1-e^{-\zeta}}\right] \tag{11}$$

As $T \to \infty$, the expression of $C^{(T)}$ can be formulated as:

$$C^{(T)} = (1-\psi)N\left[T\left((\gamma-\eta)p_d+\eta\right)-C_0\right] \tag{12}$$

$$C_0 = \eta p_0 \cdot \frac{e^{-\zeta}}{1-e^{-\zeta}} \tag{13}$$

where C_0 is a constant.

The formula indicates that as $T \to \infty$, the cumulative number of low-quality individuals exhibits a linear positive correlation with T. In the ERL framework, the experiences generated by all individuals are injected in the shared replay buffer. All individuals are assumed to generate the same number of experiences per episode. The number of low-quality individuals increases over time, leading to the accumulation of low-quality experiences in the shared replay buffer. Since the RL agent performs gradient updates based on uniformly random sampling from the buffer, the proportion of low-quality experiences in a sampled batch can be expressed as follows.

$$\rho = \frac{C^{(T)}}{NT} \tag{14}$$

$$= (1-\psi)\left[(\gamma-\eta)p_d+\eta-\eta p_0\cdot\frac{e^{-\zeta}(1-e^{-\zeta T})}{T(1-e^{-\zeta})}\right] \tag{15}$$

To better illustrate the impact of accumulated low-quality experiences on RL training, we equivalently model the sampling process as drawing experiences proportionally from separate low-quality RB_l and high-quality RB_h replay buffers and the sampling proportion of RL_{agent} experiences is temporarily excluded from consideration. In the update of the policy gradient, the gradient $\nabla_\theta J(\theta)$ can be expressed as a weighted combination of gradients calculated from each individual buffer:

$$\nabla_\theta J(\theta) = \frac{1}{|\mathcal{B}|}\sum_{s\in\mathcal{B}} \nabla_a Q_\phi(s,a)\big|_{a=\pi_\theta(s)} \cdot \nabla_\theta \pi_\theta(s) \tag{16}$$

$$= \frac{|\mathcal{B}_l|}{|\mathcal{B}|}\cdot\left(\frac{1}{|\mathcal{B}_l|}\sum_{s\in\mathcal{B}_l} \nabla_a Q_\phi(s,a)\big|_{a=\pi_\theta(s)} \cdot \nabla_\theta \pi_\theta(s)\right)$$

$$+ \frac{|\mathcal{B}_h|}{|\mathcal{B}|}\cdot\left(\frac{1}{|\mathcal{B}_h|}\sum_{s\in\mathcal{B}_h} \nabla_a Q_\phi(s,a)\big|_{a=\pi_\theta(s)} \cdot \nabla_\theta \pi_\theta(s)\right) \tag{17}$$

$$= \rho\cdot\nabla_\theta J_l(\theta) + (1-\rho)\cdot\nabla_\theta J_h(\theta) \tag{18}$$

where the $\mathcal{B}$ denotes the combined training batch, consisting of samples from two separate replay buffers. The B_l and B_h are the sub-batch sampled from the RB_l and RB_h, respectively. $Q_\phi(s,a)$ represents the estimated Q-value computed by the critic network, where ϕ denotes its parameters. $\pi_\theta(s)$ is the action output by the actor network, parameterized by θ, given state s.

ρ plays a crucial role in shaping the direction of optimization of the actor in the RL agent. A higher value of ρ increases the influence of the low-quality experience buffer, which can lead to slower convergence. When the RL agent suffers from reduced learning efficiency, it becomes less capable of generating and synchronizing high-quality policies back to the population, slowing evolutionary progress and accelerating the accumulation of low-quality experiences, which eventually triggers a performance degradation loop.

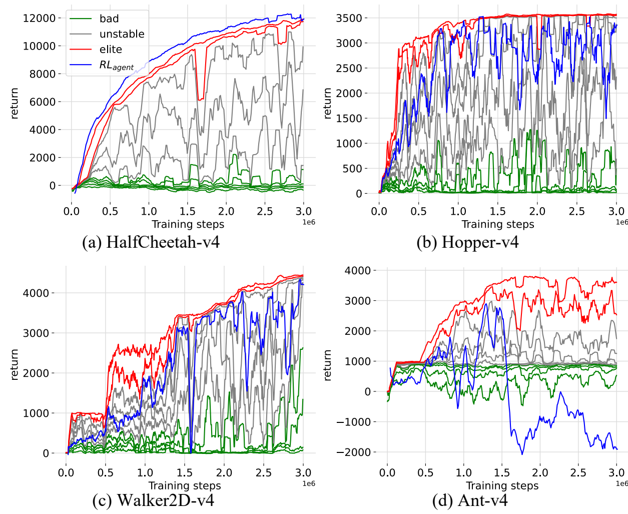


Fig. 2. The cumulative returns curves of the population (ten individuals) and the performance of RL_{agent} during the evolutionary process in PDERL. Green curves represent individuals with low-return experiences. Gray curves represent individuals with unstable experiences. Red curves represent individuals with high-return experiences. Blue curves is the performance of RL_{agent}.

Finally, we experimentally analyze the objective conflict. To gain an intuition of the impact of the objective conflict, we evaluate the quality of experiences generated by the population in PDERL and their impact on the performance of RL_{agent}, as depicted in Fig. 2. Each point is calculated by summing the rewards from interacting with the environment over a trajectory. Furthermore, all experiences from these entire trajectories are indiscriminately injected into the replay buffer. Hence, these curves reflect the quality and the experience distribution of the replay buffer and their impact on the performance of RL_{agent}.

It is observed that there is a noticeable performance gap among the individuals in the population during the training process in all environments. The performance of elite individuals exhibits relative stability. However, because of the inherent randomness, while enhancing the performance of some individuals, EA operators may also generate some bad individuals. Around half of the individuals consistently perform poorly and some unstable individuals show great performance fluctuations. These individuals persist throughout the entire training process.

The bad individuals might not have an immediate fatal impact on EAs, but they can cause prompt and severe adverse consequences in ERL algorithms. As all experiences from the population are indiscriminately injected into the replay buffer, many low-return experiences are also stored in the buffer. The increasing prevalence of low-return experiences in the buffer leads to a continuous deterioration in the quality of the buffer. Similar phenomena were also observed in the original ERL [30]. An excessive proportion of low-return experiences within the replay buffer will hamper the learning efficiency of RL_{agent}. Specifically, from Fig. 2, we can see that in Hopper and Walker2D, RL_{agent} significantly lags behind elite individuals of the population and exhibits instability. In Ant-v4, the performance of the RL_{agent} is particularly poor. Although elite individuals contribute some high-return experiences, the majority of bad and unstable individuals in the population inject more low-return experiences into the buffer. Consequently, RL_{agent} deteriorates to the most unfavorable condition and becomes irreparable, which will further seriously impact the effectiveness of synchronization.

Hence, ERL algorithms should prioritize improving the performance of all individuals in the population as much as possible, rather than solely focusing on seeking an optimal individual. Only in this way can the population generate more beneficial experiences. Consequently, the training efficiency of RL_{agent} is improved, the effect of synchronization becomes more pronounced, and the evolution of the population becomes more efficient, leading the ERL algorithms into a positive feedback loop.

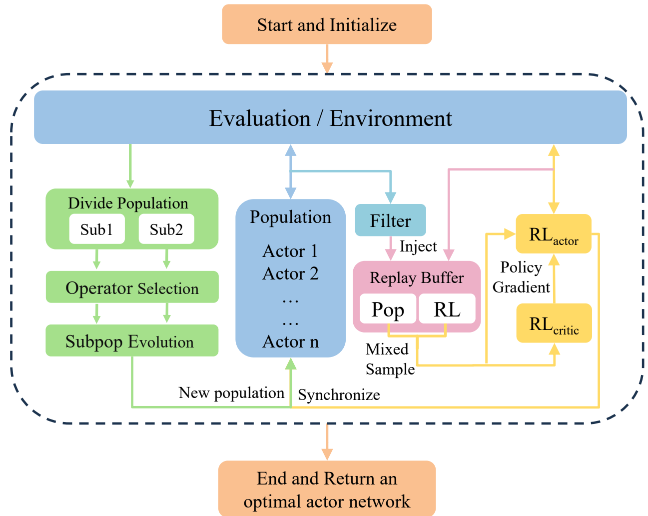


Fig. 3. Framework of SERL-OS-EF.

B. Strategic Evolutionary Reinforcement Learning with Operator Selection and Experience Filter

A diagram of SERL-OS-EF is given in Fig. 3. In the population evolution, we focus on enhancing the overall quality of the population to generate more high-quality experiences. Due to the stochastic nature of EAs, individuals will not always be good. Therefore, we divide the population into two subpopulations based on their fitness: half of the individuals with higher fitness sub_1 and the other half with lower fitness sub_2. The operator selector matches the most suitable evolutionary operator to each subpopulation based on its state. The two subpopulations evolve using their assigned operators respectively and are subsequently merged into a new population. Individuals within the population interact with the environment to generate new experiences. Before being injected into the replay buffer, these experiences must be filtered by the experience filter to maintain the long-term high quality of the replay buffer. To improve the learning efficiency of RL_{agent} and address the experience distribution mismatch, a dynamic mixed sampling strategy is proposed. RL_{agent} is optimized by the gradient updating method using mixed batches of experiences sampled from the population and RL_{agent}, with dynamic ratio adjustment throughout training. If RL_{agent} outperforms the worst individual in the population, the parameters of the actor network of RL_{agent} are synchronized with those of the worst individual.

C. Operator Selection

Our method enhances the overall quality of the population by selecting suitable EA operators for the two subpopulations at different performance levels. To adaptively determine the most suitable operators, the operator selection process is formulated as a Markov Decision Process (MDP), with an online PPO algorithm serving as the operator selector. The workflow for operator selection is shown in Fig. 4. First, the population is divided into two subpopulations based on the fitness of individuals, and the fitness values of individuals in each subpopulation are gathered. Second, metrics such as average fitness, best fitness, and the fitness difference between the previous iteration and the current iteration are calculated. Subsequently, these metrics are analyzed alongside the overall population data and historical data to assess the performance level of each subpopulation. Finally, each subpopulation is assigned the most suitable EA operator by the trained model according to its performance estimation.

To be specific, the components of the MDP, namely state space, action space, transition rule, and reward function, are defined as follows.

State: For the i-th subpopulation in the t-th iteration, its state is constructed with both its own information and the information of the entire population so that the operator selector can make more informed decision-making, i.e. $S_t^i = (S_t^{sub_i}, S_t^{pop})$. The first part is the subpopulation information, which is expressed as $S_t^{sub_i} = \{FB_t^i, FA_t^i, FBD_t^i, FAD_t^i\}$, where FB_t^i and FA_t^i denote the best fitness value and the average fitness value of the i-th subpopulations, respectively. $FBD_t^i = FB_t^i - FB_{t-1}^i$ and $FAD_t^i = FA_t^i - FA_{t-1}^i$. The second part is the information of the entire population, which is expressed as $S_t^{pop} = \{FB_t^{pop}, FA_t^{pop}\}$, where FB_t^{pop} and FA_t^{pop} represent the best fitness value and the average fitness value for the whole population. Additionally, each state feature in S_t^i is normalized within the interval [0,1] by the following equation, taking FB_t^i as an example:

$$Norm(FB_t^i) = \frac{FB_t^i - min(FB_{t-2:t}^i)}{max(FB_{t-2:t}^i) - min(FB_{t-2:t}^i)} \tag{19}$$

where $max(FB_{t-2:t}^i)$ and $min(FB_{t-2:t}^i)$ denote the maximum and minimum of FB^i over the past three iterations, respectively.

Action: The action a_t^i is defined as selecting an operator from a pre-defined set of operators for the i-th subpopulation. The operator set utilized in our method is placed at the end of this subsection.

Transition rule: The transition rule will update the current S_t^i to the next state S_{t+1}^i based on the performed action a_t^i. In our work, the transition rule is achieved by applying the

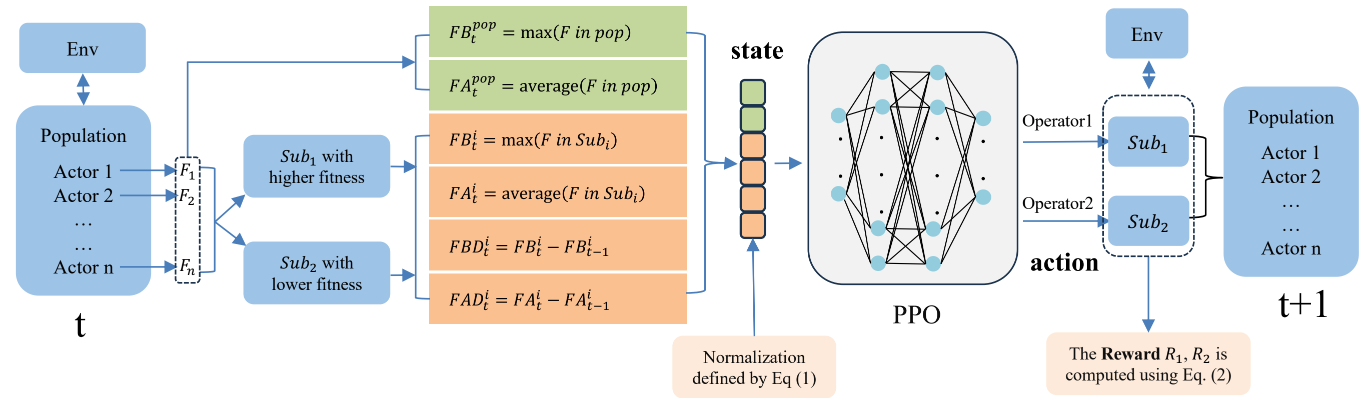


Fig. 4. The workflow of operator selection. $t, t-1, t+1$ represent the current iteration, the previous iteration, and the next iteration, respectively.

selected operators to the two subpopulations and generating a new population of the next iteration.

Reward: In existing ERL algorithms, the role of the population is often described as generating diverse experiences. However, diverse but low-return experiences offer limited assistance in the training of ERL algorithms. The population should be used to generate compatible and diverse experiences for RL_{agent}. "Compatible" refers to the notion that experiences generated by individuals in the population should match the current learning level of RL_{agent}. The experiences generated by individuals with high fitness will bring more benefits to the training of RL_{agent} compared to the experiences generated by the bad individuals. The reward function has a significant impact on the selection preference of PPO. To generate more compatible and diverse experiences, each individual in the population should exhibit good performance while still maintaining diversity. Therefore, the reward function consists of two parts: the sum of the best fitness and the average fitness $FB_t^i + FA_t^i$ and the difference between them $FB_t^i - FA_t^i$:

$$R = 0.5 * (FB_t^i + FA_t^i) + \frac{t}{T} * (FB_t^i - FA_t^i) \tag{20}$$

where T is the maximum number of iterations.

In the first part of the reward function $FB_t^i + FA_t^i$, both the best fitness FB_t^i and the average fitness FA_t^i have equal significance, which contributes to improving the performance of each individual, rather than just improving the performance of elite individuals. The second part of the reward function $FB_t^i - FA_t^i$ is designed to indirectly reflect the diversity and heterogeneity among individuals in the subpopulation. Regarding the diversity of the population, an intuitive measurement from the phenotype perspective is the standard deviation of the fitness values of all individuals. For the sake of simplicity, we use $FB_t^i - FA_t^i$ as an approximation. In the initial stage of the algorithm, individuals in the population are randomly generated. The population is naturally very diverse. At this time, t is very small, and the second part of (20) does not play a key role in the reward of operator selection. Along with the training process, normally the algorithm would converge and the diversity of the population would gradually decrease. Hence, the weight of the second part increases as the iteration progresses (t increases), which makes the algorithm focus more and more on maintaining the diversity of the population, contributing to fostering the generation of compatible and diverse experiences.

Operator Set: In this work, two efficient and safe operators, namely, opposite-based proximal mutation and gradient optimization operator, are employed to construct the operator set.

1) Opposite-based proximal mutation operator. Mutation typically exhibits significant randomness and is prone to catastrophic forgetting. Although proximal mutation [23] mitigates this issue to some extent by controlling the mutation magnitude through sensitivity calculation, it still retains substantial randomness. Therefore, we further propose the opposite-based proximal mutation to enhance the robustness and performance of mutation by refining the adjustment of the mutation magnitude. Firstly, we perform a basic mutation operation. Sensitivity s_e, which is defined by (21), is utilized to adjust the Gaussian perturbation of each weight. The sensitivity s is calculated by the gradient of each dimension of the output action over N_M transitions, which are sampled from the buffer of individuals.

$$s_e = \sqrt{\sum_k^A \left(\sum_i^{N_M} \nabla_\theta \mu_\theta(s_i)_k\right)^2} \tag{21}$$

$$\theta \leftarrow \theta + \frac{x}{s_e} \tag{22}$$

where $x \sim N(0, \sigma I)$. θ represents the parameters of actor-network μ. σ represents the mutation magnitude parameter. A denotes the dimension of the output action.

Secondly, we perform an opposite mutation to adjust the mutation magnitude further. A scaling factor α is designed to adaptively adjust the scale of the opposite mutation based on the performance of the basic mutation.

$$\theta \leftarrow \theta - \alpha * \frac{x}{s_e}$$
$$\alpha = abs(\frac{F_a}{F_b}) \tag{23}$$

where the $abs()$ function returns the absolute value of the number. F_a and F_b are the fitness values before and after the basic mutation, respectively.

Finally, The fitness of the opposite mutation is compared with that of the basic mutation. The mutation with better fitness is adopted.

2) Gradient optimization operator: the critic network of RL_{agent} is used to train the actor networks of individuals in the population by the sampled policy gradient. If the gradient optimization operator is selected, each individual in the population is subject to a probability 90% of undergoing 100 rounds of gradient training. The probability is the same as the probability of the proximal mutation in PDERL.

D. Experience Filter

Despite extensive efforts to improve the performance of all individuals in the population, complete eradication of bad individuals remains impossible due to the stochasticity of EAs. To maintain the long-term high quality of the replay buffer, it is essential to regulate the experiences being injected, replacing the indiscriminate experience storage mechanism in the existing ERL algorithms. Inspired by the move-acceptance strategy [39], [40], the experience filter strategy is proposed to retain high-quality experiences while discarding low-quality ones. The core of this mechanism lies in how to evaluate the quality of experiences. TD-error is used to measure the importance of each transition in prioritized experience replay [41]. Despite its advantages, the prioritizing experience framework involves some intricacy. To simplify the complexity and reduce computational costs, we use a coarse-grained measurement that the fitness of individuals is established as the criterion for deciding whether to discard or retain experiences provided by individuals in the population. It works at a trajectory level instead of a single experience level.

In the experience filter, the experiences generated by the population are expected to be compatible with RL_{agent}. If an individual exhibits a significant gap in fitness to RL_{agent} in an iteration, all experiences generated by the individual in this iteration will be discarded. Due to the comparatively lower stability of RL_{agent}, elite individuals in the population are employed as substitutes. The pseudocode of the experience filter is shown in Algorithm 1, where β is a parameter used to adjust the filtering intensity.

E. Dynamic Mixed Sampling Strategy

Experience collection and utilization are crucial to RL, and an efficient experience replay strategy can significantly enhance the learning efficiency and overall performance of the algorithm [42]. In existing ERL algorithms, the experiences generated by the population and RL_{agent} are merged into the same shared replay buffer, associated with a uniform random sampling strategy for the gradient-based optimization of RL_{agent} training. However, there is an experience distribution mismatch between the experiences generated by the population and RL_{agent} [30]. To address this issue, a dynamic mixed sampling strategy is proposed. The experiences generated by

Algorithm 1 Experience Filter

Input: Population size N, the fitness values of population $F_i, i = 1, 2, ...N$, the best fitness value of the population BF, filter parameter β, the trajectories of all individuals in the population, the replay buffer of the population RB_{pop};

```
for i = 1 → N do
    if BF > 0 & F_i > BF * (1 − β), 0 < β then
        the trajectory from individual i is added to RB_pop;
    else
        the trajectory from individual i is discarded;
    end if
    if BF < 0 & F_i > BF * (1 + β) then
        the trajectory from individual i is added to RB_pop;
    else
        the trajectory from individual i is discarded;
    end if
end for
return ;
```

the population and RL_{agent} are stored separately in different replay buffers, and one batch is generated according to:

$$\hat{d}(s,a) = m * d_{D_{rl}}(s,a) + (1-m) * d_{D_{pop}}(s,a) \quad (24)$$

where $d_{D_{rl}}(s,a)$ and $d_{D_{pop}}(s,a)$ are batches separately sampled from the buffers of RL_{agent} and the population. $\hat{d}(s,a)$ is a mixed final batch. m represents the proportion of experiences sampled from the buffers of RL_{agent}.

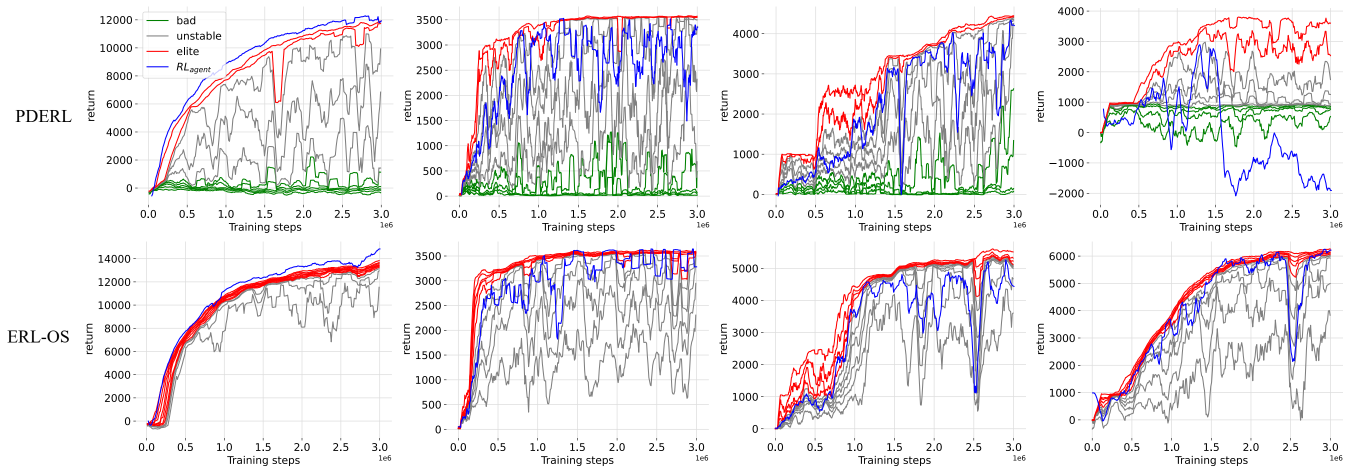
Due to the implementation of the experience filter, the compatible and diverse experiences generated by the population will contribute positively to the learning efficiency of RL_{agent}. However, sampling too many population experiences may introduce instability to the learning of RL_{agent}. In the early stages, more experiences should be sampled from the buffer of RL_{agent}, which contributes to the rapid and stable learning of RL_{agent}. In the later stages, the diverse experiences in the buffer of the population benefit the further enhancement of RL_{agent}. Therefore, a linearly decreasing ratio is used to balance the experience proportion from RL_{agent} and the population:

$$m = k * (1 - \frac{t}{T}) \quad (25)$$

where k is a hyperparameter.

IV. Experimental Study

In this section, we provide details of the experimental setup in the first place. Then, the components of SERL-OS-EF including operator selection, experience filter, and dynamic mixed sampling strategy are validated and tuned in four MoJoCo environments. Finally, the effectiveness, efficiency, and practical significance of SERL-OS-EF are evaluated in four MuJoCo environments, three Ant-Maze environments with deceptive rewards, and a practical task.



(a) HalfCheetah-v4 (b) Hopper-v4 (c) Walker2D-v4 (d) Ant-v4

Fig. 5. The comparison between PDERL (top row) and ERL-OS (bottom row) regarding the quality of individuals in the population.

A. Experimental Setup

To mitigate the critical overestimation issue in the RL part, TD3 is adopted as the RL_{agent} in our algorithm SERL-OS-EF. For a fair comparison, PDERL also replaces DDPG with TD3. The original network parameters of TD3 are used in SERL-OS-EF and PDERL. The parameter settings of our method are present in Table III and IV.

TABLE III
PARAMETER SETTING

Parameter	Value	Ref
The parameter of TD3	default setting	[43]
The size of population buffer and RL_{agent} buffer	50,000	[30]
The size of indivudual buffer in population	8,000	[23]
The size of population	10	[18], [23]
The number of elite individuals	2	[18], [23]
The Mutation magnitude parameter σ	0.01	[23]
The batch size N_M	256	[23]
The scaling factor α	1.5	Ours
The parameter of filter β	0.25	Ours
The parameter of dynamic mixed sampling k	0.2	Ours

TABLE IV
PPO PARAMETER SETTINGS

Hyperparameter	Value
Actor network	FC(64,32)
Actor activate function	ReLU
Critic network	FC(64,32)
Critic activate function	ReLU
Optimizer	Adam
Discount factor	0.99
Clip range	0.2
Learning rate	$3 \cdot 10^{-4}$
Advantage estimation parameter	0.95
Number of training epochs per update	4
Batch size	5
Number of steps per update	20

B. Investigation of Operator Selection

1) Visualization of Individual Quality: To validate the effectiveness of the proposed operator selection strategy, we simplify our algorithm and keep only the operator selection part. The simplified version is denoted as ERL-OS. First, we demonstrate that ERL-OS can generate more beneficial experiences by conducting the same experiment we have done on PDERL with the same random seeds. The results on ERL-OS are shown in Fig. 5, compared with PDERL. It is observed that the performance of individuals in ERL-OS greatly outperforms that of PDERL. In the population of ERL-OS, the type of bad individual (green curves) has disappeared in all environments. The number of elite individuals in the population has grown, and the fluctuations of unstable individuals are more subdued compared to those in PDERL, especially in HalfCheetah. Meanwhile, the noticeable gap between the worst and elite individuals has decreased in ERL-OS, which leads to superior experiences being injected into the replay buffer. Hence, there is a notable improvement in the final performance and stability of RL_{agent} in all environments. The results indicate that the proposed ERL-OS effectively enhances the quality of population experiences and the learning of RL_{agent} benefits from these experiences.

2) Comparison with only one operator: To further figur out whether the selection strategy is useful or only the proposed mutation operator is useful, we designed two other ERL algorithms, GERL and OPERL, using only the gradient optimization operator and the opposite-based proximal mutation operator, respectively. ERL-OS is compared with them. Meanwhile, to investigate the relationship between the performance of RL_{agent} and ERL algorithms, the learning curves of three ERL algorithms (solid line) and their corresponding RL_{agent} (dashed line) are illustrated in Fig. 6. From a holistic perspective, ERL-OS consistently outperforms GERL and OPERL throughout the entire iteration process, indicating the beneficial impact of operator selection on enhancing the performance of ERL-OS.

In the previous comparison, we have witnessed the impact of population on RL_{agent}. Here, we will further analyze the influence of RL_{agent} on the ERL algorithms. These analyses aim to offer a comprehensive understanding of the symbiotic relationship between evolution and learning. As shown in Fig. 6, the performance of RL_{agent} shows a positive correlation with the algorithm's effectiveness. The better the performance of RL_{agent}, the better the performance of ERL algorithms. RL_{agent} contributes to an ERL algorithm in two ways. The first way involves directly boosting the algorithmic performance, where the current performance of RL_{agent} surpasses the historical best value of the algorithm. In the initial to middle phases of HalfCheetah, the curves of ERL algorithms closely follow those of RL_{agent}, indicating that RL_{agent} directly enhances the performance of ERL algorithms through synchronization. The actor network of RL_{agent} is synchronized to the population and retained as an elite individual. Another way is to indirectly improve the performance of the ERL algorithms, where RL_{agent} provides a promising exploration space for population evolution. In most cases, the second way is more prevalent. In the early stages of Hopper and Walker2D, the performance improvement of RL_{agent} in ERL-OS is faster than that in GERL and OPERL, leading to superior convergence speed in ERL-OS. In the later stage of HalfCheetah and the middle to late stage of Ant, better RL_{agent} contributes to ERL-OS achieving better performance.

Table V reports the final performance (Mean±Std.) of OPERL, GERL, and ERL-OS. The result with the highest mean value is shown in bold. Combining Fig. 6 and Table V, it is evident that ERL-OS not only converges faster than the other two algorithms but also outperforms them significantly in the final performance. The experimental results from Fig. 5 and Fig. 6 demonstrate that the operator selection strategy contributes to elevating the quality of population experiences, which facilitates the improvement of RL_{agent}. The improved RL_{agent} will further enhance the efficiency of the population, which propels ERL-OS into a positive feedback loop. Ultimately, the overall performance of ERL-OS is improved.

TABLE V
THE FINAL PERFORMANCE (MEAN±STD.) OF OPERL, GERL, AND ERL-OS

Algorithm	HalfCheetah	Hopper	Walker2D	Ant
OPERL	13881±754	3709±60	5230±477	5648±763
GERL	13714±415	3749±64	5183±434	5565±715
ERL-OS	**14988±868**	**3765±30**	**5540±371**	**6207±755**

C. Investigation of Experience Filter and Dynamic Mixed Sampling Strategy

In the experience filter and dynamic mixed sampling strategies, there are two key parameters: filtering parameter β and mixed sampling parameter k. We employed a controlled variable method to evaluate their influence on SERL-OS-EF independently. Firstly, β is set to 0.25, while $k = 0.3, 0.2, 0.1$. The learning curves of ERL-OS and SERL-OS-EF with different values of k are shown in Fig. 7. The final performance is presented in Table VI. The result with the highest mean value is highlighted. The larger the value of k, the more experiences are sampled from the buffer of RL_{agent}, and the fewer experiences are sampled from the buffer of the population.

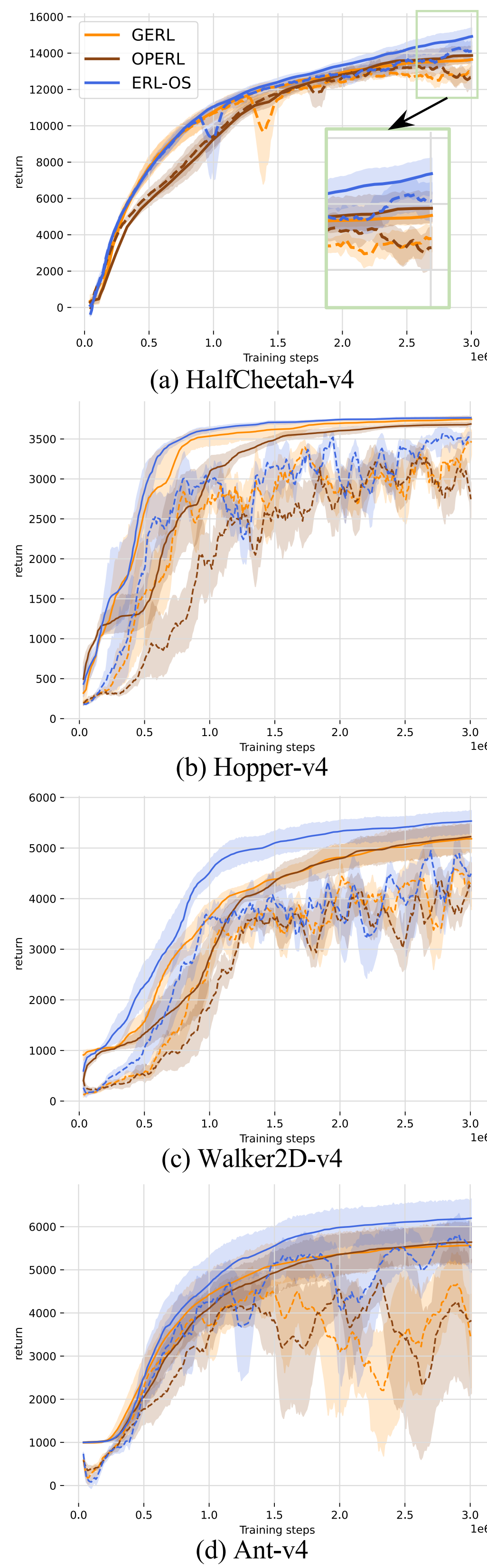


Fig. 6. The learning curves (solid line) of GERL, OPERL, and ERL-OS during the evolutionary process. The learning curves of the RL_{agent} are drawn by dash lines with the same color

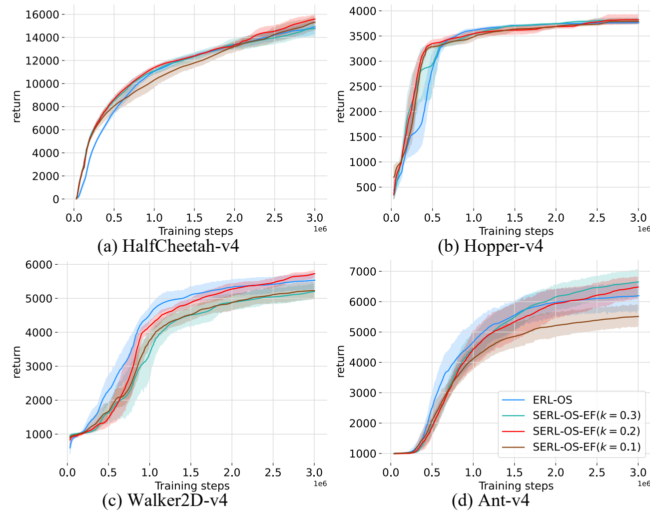


Fig. 7. the learning curves of ERL-OS and SERL-OS-EF with different value of k.

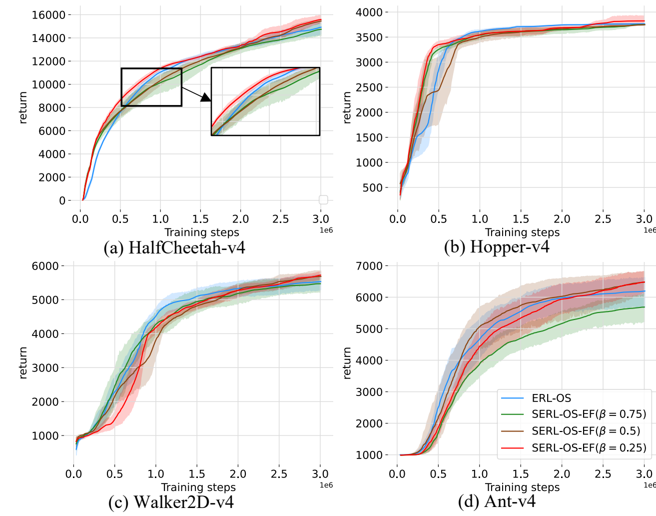


Fig. 8. The learning curves of ERL-OS and SERL-OS-EF with different value of β.

TABLE VI
THE FINAL PERFORMANCE (MEAN±STD.) OF ERL-OS AND SERL-OS-EF WITH DIFFERENT VALUE OF k.

Algorithm	HalfCheetah	Hopper	Walker2D	Ant
ERL-OS	14988±868	3765±30	5540±371	6207±755
SERL-OS-EF $k = 0.3$	14766±1159	3820±60	5206±331	**6666±727**
SERL-OS-EF $k = 0.2$	**15610±584**	**3825±151**	**5736±173**	6500±542
SERL-OS-EF $k = 0.1$	15388±991	3790±79	5227±396	5533±542

TABLE VII
THE FINAL PERFORMANCE (MEAN±STD.) OF ERL-OS AND SERL-OS-EF WITH DIFFERENT VALUE OF β.

Algorithm	HalfCheetah	Hopper	Walker2D	Ant
ERL-OS	14988±868	3765±30	5540±371	6207±755
SERL-OS-EF $\beta = 0.25$	**15610±584**	**3825±151**	**5736±173**	**6500±542**
SERL-OS-EF $\beta = 0.5$	15521±317	3749±26	5689±292	6487±628
SERL-OS-EF $\beta = 0.75$	14769±841	3751±13	5473±393	5691±708

As shown in Fig. 7, in the early stages of HalfCheetah and Hopper, the differences among SERL-OS-EF with varying k values are minimal, yet all demonstrate superior performance compared to ERL-OS. In Walker2D and Ant, ERL-OS shows a faster convergence speed initially, but SERL-OS-EF($k = 0.2$) catches up eventually and SERL-OS-EF($k = 0.3$) achieves the best final performance. The results suggest that different values of k have varying effects on SERL-OS-EF in different environments. According to Fig. 7 and Table VI, a large value of k is advantageous for Hopper and Ant. A small value of k is beneficial for HalfCheetah. However, both maximum and minimum values of k result in decreased performance of SERL-OS-EF. Hence, there is a trade-off regarding the setting of k. In terms of final performance, SEH-OS-EF($k = 0.2$) ranks first on three out of four environments and ranks second in Ant. Hence, the $k = 0.2$ is chosen as the default parameter.

Secondly, to evaluate the influence of β on the SERL-OS-EF, k is set to 0.2, while $\beta = 0.75, 0.5, 0.25$. The learning curves of ERL-OS and SERL-OS-EF with different values of β are plotted in Fig. 8. The final performance is listed in Table VII. The results with the highest mean values are highlighted. The β denotes the tolerance level of the experience filter towards experiences generated by subpar individuals. A smaller value of β indicates a smaller tolerance threshold, which will filter out more low-quality experiences.

As shown in Fig. 8, in the early stage of HalfCheetah, the learning curves of SERL-OS-EF($\beta = 0.75$) and SERL-OS-EF($\beta = 0.5$) are nearly overlapped. This phenomenon stems from the minor discrepancies among individuals in the population. There are no individuals that are significantly inferior to the best one. Consequently, the experience filters of SERL-OS-EF($\beta = 0.75$) and SERL-OS-EF($\beta = 0.5$) remain inactive. In the early stages of Walker2D and Ant, the convergence speed of SERL-OS-EF($\beta = 0.25$) is lower than that of ERL-OS, indicating that filtering out too many low-reward experiences may have a negative impact on SERL-OS-EF. In the mid to later stages of Walker2D and Ant, the performance of ERL-OS shows a slow improvement, while the SERL-OS-EF($\beta = 0.25$) exhibits a notable and consistent enhancement during the same period. SERL-OS-EF($\beta = 0.25$) achieves the best final performance in all environments, as presented in Table VII. Hence, $\beta = 0.25$ is selected as the default parameter. To sum up, the default parameters for SERL-OS-EF are as follows: $k = 0.2, \beta = 0.25$.

D. SERL-OS-EF Evaluation

To evaluate the overall performance of the proposed SERL-OS-EF, we compare it with three DRL algorithms, namely DDPG, TD3, and SAC, and six state-of-the-art ERL algo-

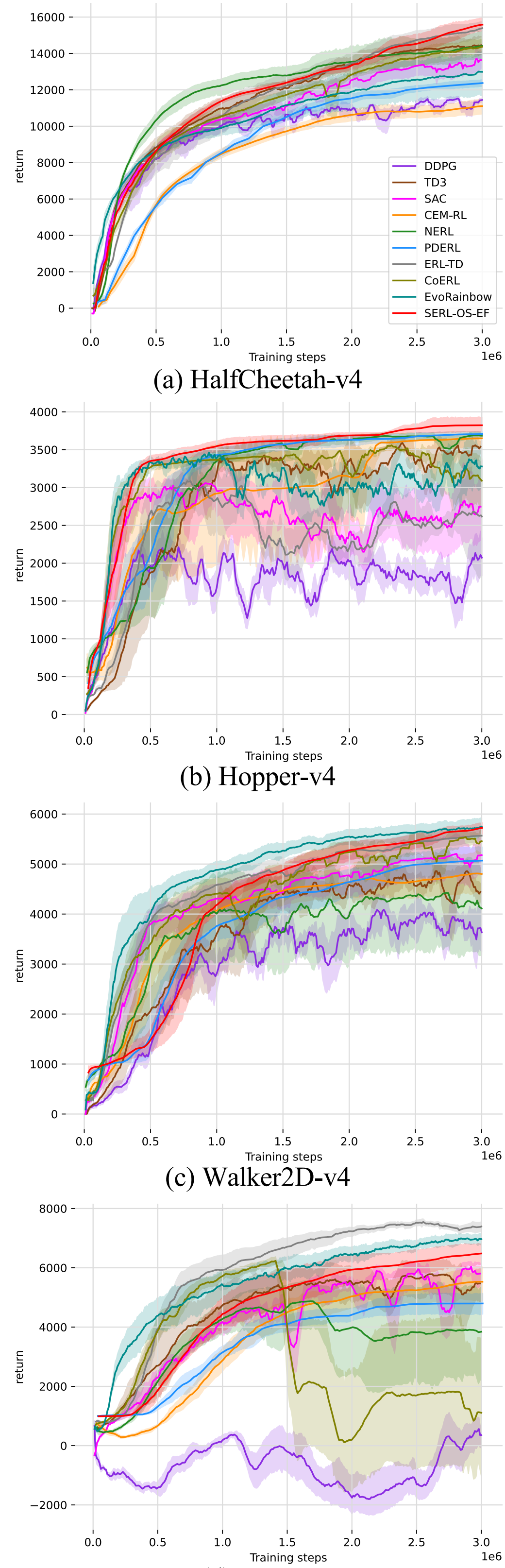
TABLE VIII
THE FINAL PERFORMANCE (MEAN±STD.(RANK)) OF ALL ALGORITHMS ON MUJOCO ENVIRONMENTS.

Algorithm	HalfCheetah	Hopper	Walker2D	Ant	Rank
DDPG	11766±335(9)	1948±512(10)	3172±1346(10)	-538±1190(10)	9.75
TD3	14631±816(3)	3620±111(5)	4762±494(8)	5847±975(4)	5
SAC	13948±871(6)	2813±875(8)	5300±660(5)	5793±754(5)	6
CEM-RL	11143±730(10)	3655±155(4)	4787±681(7)	5573±816(6)	6.75
NERL	14394±1302(4)	3671±73(3)	4084±1371(9)	4037±2391(8)	6
PDERL	12436±261(8)	3708±50(2)	5073±427(6)	4800±549(7)	5.75
ERLTD	15390±660(2)	2617±1194(9)	5567±177(3)	**7392±284(1)**	3.75
CoERL	14369±617(5)	3090±474(7)	5456±346(4)	1109±3429(9)	6.25
EvoRainbow	12989±1871(7)	3278±439(6)	**5748±281**(1)	6967±297(2)	4
SERL-OS-EF	**15610±584**(1)	**3825±151**(1)	5736±173(2)	6500±542(3)	1.75

rithms, namely CEM-RL[1], NERL[2], PDERL[3], ERL-TD[4], CoERL[5], and EvoRainbow [44] [6] on four MuJoCo environments and three Ant-Maze environments with deceptive rewards [7]. For a fair comparison, the proximal mutation in PDERL has been replaced by the opposite-based proximal mutation and the synchronization aligns with that of SERL-OS-EF.

1) MuJoCo environments: HalfCheetah, Hopper, Walker2D, and Ant environments involve controlling different types of physical agents and are widely used benchmarks for evaluating reinforcement learning algorithms. The learning curves are illustrated in Fig. 9. The final results are presented in Table VIII. Table VIII shows that the SERL-OS-EF achieved the best average rank among all methods in four environments, demonstrating its superior effectiveness. In HalfCheetah and Walker2D, some algorithms initially converge faster than SERL-OS-EF, but their convergence rate has shown a decelerating trend, suggesting that they may have converged to a local optimum. However, SERL-OS-EF maintains a stable and faster convergence rate in the middle to late stages. Ultimately, SERL-OS-EF catches up with other algorithms in Walker-2D and outperforms them in HalfCheetah. In Hopper, SERL-OS-EF consistently shows superior performance compared to other algorithms. Although reaching a local optimum similar to other algorithms in the mid-term, SERL-OS-EF continues to make further improvements in the later stages. In Ant, ERL-TD significantly outperformed other algorithms, primarily due to the use of multiple critic networks and a truncated variance strategy to mitigate overestimation bias. However, ERL-TD incurs significantly higher computational costs. As shown in Table IX, its runtime of training 10,000 steps in the four MuJoCo environments is quadruple that of our algorithm. CoERL converges quickly in the early stages but shows significant performance drops in the later stages of Ant, revealing the risk of evolving the actor network of RL_{agent} through cooperative coevolution. Although EvoRainbow performed well in the early stages across all four environments, its performance declined in the later stages on HalfCheetah and Hopper, and its computational cost was over twice that of ours.

[1]https://github.com/apourchot/CEM-RL.
[2]https://github.com/cugqiaorui/nerl.
[3]https://github.com/crisbodnar/pderl.
[4]https://github.com/2019cyf/ERL-TD
[5]https://github.com/HcPlu/CoERL
[6]https://github.com/yeshenpy/EvoRainbow
[7]https://github.com/Farama-Foundation/Gymnasium-Robotics?tab=readme-ov-file



(a) HalfCheetah-v4
(b) Hopper-v4
(c) Walker2D-v4
(d) Ant-v4

Fig. 9. Learning curves of all algorithms on MuJoCo environments.

TABLE IX
RUNTIME (IN SECONDS) OF TRAINING 10,000 STEPS ON MUJOCO ENVIRONMENTS.

Algorithm	HalfCheetah	Hopper	Walker2D	Ant
DDPG	169.71	170.23	170.36	177.91
TD3	140.68	143.10	141.96	146.10
SAC	381.83	385.00	399.16	404.74
CEM-RL	340.45	298.94	336.93	354.19
NERL	335.05	357.71	313.41	351.47
PDERL	196.85	158.89	176.81	182.81
ERL-TD	935.84	910.57	1278.32	977.47
CoERL	403.40	396.69	434.29	342.69
EvoRainbow	557.42	593.79	558.88	580.94
SERL-OS-EF	188.66	229.38	232.57	224.41

TABLE X
THE PERFORMANCE (MEAN(MAXIMUM, MEDIAN)) OF ALL ALGORITHMS ON ANT-MAZE ENVIRONMENTS.

Algorithm	Ant-UMaze	Ant-MediumMaze	Ant-LargeMaze
DDPG	3.09(14.71, 0.24)	0.51(1.99, 0)	1.32(6.58, 0)
TD3	38.86(190.91, 0.77)	4.44(16.35, 0)	16.61(83.05, 0)
SAC	2.95(13.95, 0.30)	0.53(2.02, 0)	1.52(7.58, 0)
CEM-RL	109.17(209.07, 152.63)	6.75(25.08, 0)	30.90(154.52, 0)
NERL	34.26(90.25, 34.13)	6.26(13.60, 4.34)	23.45(117.25, 0)
PDERL	270.42(387.36, 329.89)	163.55(507.43, 108.55)	132.84(665.07, 0)
ERL-TD	64.91(203.47, 51.68)	19.69(84.01, 0.01)	53.30(266.51, 0)
CoERL	2.85(13.45, 0.30)	0.51(1.99, 0)	1.35(6.75, 0)
EvoRainbow	21.06(60.23, 6.95)	18.30(60.32, 0.04)	26.51(132.55, 0)
SERL-OS-EF	**393.21(583.93, 410.96)**	**320.81(586.17, 240.93)**	**267.85(775.23, 181,31)**

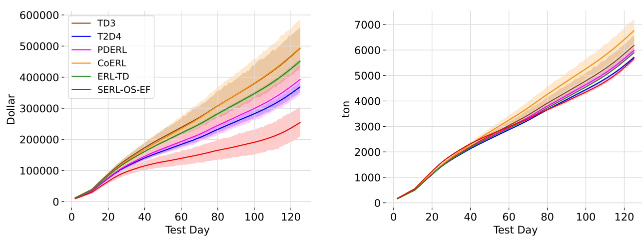
2) Ant-Maze environments with deceptive rewards: This benchmark contains three environments, Ant-UMaze, Ant-MediumMaze, and Ant-LargeMaze. The complexity of the three maze environments increases in succession. These environments aim to guide an ant agent through the maze to reach the target point. The reward signal is deceptive, defined as the negative Euclidean distance between the ant and the target point, with a negative exponential function applied to calculate the reward value. The closer the ant is to the target point, the higher the reward. The simultaneous demands of navigation and movement control make this reward signal prone to causing collisions or getting the ant stuck.

Table X presents the Mean(Maximum, Median) values across 5 seeds for all algorithms. DRL algorithms such as DDPG, TD3, and SAC exhibit poor performance in deceptive environments. This is primarily due to their insufficient exploration capability, making it challenging for them to effectively learn from deceptive rewards. Both ERL-TD and CoERL show poor performance, primarily due to their compromise on exploration. ERL-TD emphasizes improving the RL component, while CoERL optimizes the same actor network for both the EA and RL components. Consequently, both algorithms reduce the exploration ability of EA components, which is the key reason for their poor performance. Although EvoRainbow integrates five different mechanisms, it still underperforms in complex maze environments. CEM-RL achieves promising results on some seeds in the Ant-UMaze. However, its median values drop to zero in the Ant-MediumMaze and Ant-LargeMaze, indicating a significant lack of learning capability under more complex maze environments. PDERL demonstrates relatively strong performance in Ant-UMaze and Ant-MediumMaze. However, its median also drops to 0 in Ant-LargeMaze, indicating a significant decline in performance. Our method places greater emphasis on the quality of the population and the synergy between the EA and RL components, rather than solely enhancing one side's capability. As a result, it effectively balances exploration and exploitation. SERL-OS-EF algorithm achieves the best performance and demonstrates greater stability in performance across all Ant-Maze environments, highlighting the effectiveness of our method.

E. Low-carbon Multi-energy Microgrid Energy Management Task

To demonstrate the practical applicability of the proposed method, we test SERL-OS-EF on a low-carbon multi-energy microgrid energy management task [45]. The efficient deployment of renewable energy in power systems is a key strategy for achieving carbon peaking and neutrality, as well as mitigating the environmental and carbon emission pressures caused by fossil energy. A multi-energy microgrid (MEMG) system is a small-scale energy system that combines various distributed energy resources, including combined heat and power, gas boiler, electric energy system, thermal energy system, renewable energy resources like photovoltaic and wind turbines, along with numerous flexible loads. Optimally scheduled multi-energy flows in MEMG are desirable to reduce system costs and carbon emissions. To reduce overall system-wide carbon emissions and promote the adoption of renewable energy, carbon emission trading, integral carbon price model, and green certificate market mechanisms are introduced in [45]. Meanwhile, to address the challenges of dynamism, uncertainty, and scalability in MEMG systems, [45] models the task as a Markov Decision Process using the publicly available dataset. Specifically, the electric and photovoltaic profiles are sourced from the Ausgrid dataset [46], while the thermal load profile is obtained from the dataset provided in [47]. The entire dataset is available at an hourly granularity for a full year and divided into training and testing sets. For each month, the first 20 days are used for training and the rest for testing, resulting in 240 training days and 125 testing days annually. For consistency, the experimental setup is aligned with those presented in [45], to which the reader is referred for further information.

A two-step diffusion policy TD3 (T2D4) algorithm is proposed in [45], which is the state-of-the-art RL algorithm for low-carbon multi-energy microgrid energy management task. T2D4 combines TD3 with a diffusion model, which is used to fit the true distribution of history data under various uncertainties. In addition to T2D4, we also include TD3, PDERL, CoERL and ERL-TD as the benchmark algorithms for comparison. All algorithms are trained for 20,000 episodes (aligned with the setup in [45]) on the training dataset and subsequently evaluated on the testing dataset. The accumulated cost and carbon emission over test days is plotted in Fig. 10. The comparison results are shown in Table XI. PDERL and ERL-TD demonstrate superior performance over TD3 in reducing both cost and carbon emissions, although they do not outperform T2D4. Moreover, ERL-TD exhibits the lowest variance, indicating superior stability. In contrast, CoERL shows slightly lower performance than TD3, possibly due to the limitations of its cooperative coevolution mechanism under more complex practical task. SERL-OS-EF achieves the lowest cumulative cost and carbon emissions among all methods. Specifically, SERL-OS-EF achieves 30.20%–47.66% lower costs and 0.13%–15.35% lower carbon emissions compared to the other algorithms. Overall, the experimental results demonstrated the efficacy and practicality of our method in the practical task.



(a) Cost (b) Carbon emission

Fig. 10. The cumulative cost and carbon emission of the whole test days. The lower the better.

TABLE XI
COMPARISON ON THE MEMG TASK.

Algorithm	Cost(thous.$)	Carbon emission(ton)
TD3	516.26±108.75	6478.54±555.41
T2D4	387.73±247.58	5987.44±53.16
PDERL	412.59±76.18	6279.51±174.26
CoERL	517.14±153.68	7064.44±739.17
ERL-TD	472.90±8.94	6182.24±12.95
SERL-OS-EF	**270.65±80.59**	**5979.91±115.63**

V. CONCLUSION

The goal of this paper is to reveal the issues in the process of experience generation and utilization in ERL and to enhance the performance of ERL by addressing them. Through analyzing the quality differences among individuals during the evolutionary process, we have successfully identified the issue as the conflict of objectives between population evolution in EA and ERL. To address this issue, SERL-OS-EF was proposed to enhance the synergy by improving the quality of the shared replay buffer and maximizing its utility. Firstly, the operator selection strategy was proposed to boost the production of compatible and diverse individuals, fundamentally improving the quality of experiences flowing into the replay buffer. Secondly, an experience filter was proposed to filter out the experiences generated by poor individuals, which facilitates the maintenance of a long-term high-quality replay buffer. Finally, a dynamic mixed sampling strategy is introduced to enhance the efficiency of RL_{agent} in learning from the buffer, thereby improving overall synergy efficiency. The superiority of SERL-OS-EF is demonstrated through four MuJoCo environments, three Ant-Maze environments with deceptive rewards, and a low-carbon multi-energy microgrid energy management task.

In future works, there are three directions in which we will continue our research on ERL. First, we will delve deeper into the synergy between the EA and RL components in ERL algorithms. Second, we aim to enhance the performance and robustness of ERL algorithms in environments with deceptive rewards. Finally, we will apply ERL algorithms to more real-world applications like vehicle routing problems and multi-agent systems.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[2] 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*, vol. 518, no. 7540, pp. 529–533, 2015.
[3] M. Han, K. Wong, J. Euler-Rolle, L. Zhang, and R. K. Katzschmann, "Robust learning-based control for uncertain nonlinear systems with validation on a soft robot," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
[4] T. Zhang, L. Yue, C. Wang, J. Sun, S. Zhang, A. Wei, and G. Xie, "Leveraging imitation learning on pose regulation problem of a robotic fish," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[5] Y. Yang, B. Kiumarsi, H. Modares, and C. Xu, "Model-free λ-policy iteration for discrete-time linear quadratic regulation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 635–649, 2021.
[6] Y. Yang, Y. Pan, C.-Z. Xu, and D. C. Wunsch, "Hamiltonian-driven adaptive dynamic programming with efficient experience replay," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[7] Z. Bing, H. Zhou, R. Li, X. Su, F. O. Morin, K. Huang, and A. Knoll, "Solving robotic manipulation with sparse reward reinforcement learning via graph-based diversity and proximity," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 3, pp. 2759–2769, 2022.
[8] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287.
[9] G. Wang, M. Xin, W. Wu, Z. Liu, and H. Wang, "Learning of long-horizon sparse-reward robotic manipulator tasks with base controllers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 3, pp. 4072–4081, 2022.
[10] X. Wang, Y. Laili, L. Zhang, and Y. Liu, "Hybrid task scheduling in cloud manufacturing with sparse-reward deep reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, 2024.
[11] M. Pecháč, M. Chovanec, and I. Farkaš, "Self-supervised network distillation: An effective approach to exploration in sparse reward environments," *Neurocomputing*, p. 128033, 2024.
[12] Z. Fang, B. Zhao, and G. Liu, "Image augmentation based momentum memory intrinsic reward for sparse reward visual scenes," *IEEE Transactions on Games*, 2023.
[13] A. Y. Majid, S. Saaybi, V. Francois-Lavet, R. V. Prasad, and C. Verhoeven, "Deep reinforcement learning versus evolution strategies: a comparative survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
[14] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.
[15] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *arXiv preprint arXiv:1206.4621*, 2012.
[16] 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.
[17] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.
[18] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[19] Y. Shao, J. C.-W. Lin, G. Srivastava, D. Guo, H. Zhang, H. Yi, and A. Jolfaei, "Multi-objective neural evolutionary algorithm for combinatorial optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[20] W. Long, T. Hou, X. Wei, S. Yan, P. Zhai, and L. Zhang, "A survey on population-based deep reinforcement learning," *Mathematics*, vol. 11, no. 10, p. 2234, 2023.

[21] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein, "Guided evolutionary strategies: Augmenting random search with surrogate gradients," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4264–4273.

[22] H. Qian and Y. Yu, "Derivative-free reinforcement learning: a review," *Frontiers of Computer Science*, vol. 15, no. 6, p. 156336, 2021.

[23] C. Bodnar, B. Day, and P. Lió, "Proximal distilled evolutionary reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3283–3290.

[24] Q. Lin, Y. Chen, L. Ma, W.-N. Chen, and J. Li, "Erl-td: Evolutionary reinforcement learning enhanced with truncated variance and distillation mutation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, 2024, pp. 13 826–13 836.

[25] A. Pourchot and O. Sigaud, "Cem-rl: Combining evolutionary and gradient-based methods for policy search," *arXiv preprint arXiv:1810.01222*, 2018.

[26] C. Hu, R. Qiao, W. Gong, X. Yan, and L. Wang, "A novelty-search-based evolutionary reinforcement learning algorithm for continuous optimization problems," *Memetic Computing*, vol. 14, no. 4, pp. 451–460, 2022.

[27] P. Li, H. Tang, J. Hao, Y. Zheng, X. Fu, and Z. Meng, "ERL-Re^2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation," in *International Conference on Learning Representations*, 2023.

[28] C. Hu, J. Liu, and X. Yao, "Evolutionary reinforcement learning via cooperative coevolution," in *European Conference on Artificial Intelligence*, 2024.

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

[30] B. Zheng and R. Cheng, "Rethinking population-assisted off-policy reinforcement learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 624–632.

[31] O. Sigaud, "Combining evolution and deep reinforcement learning for policy search: a survey," *ACM Transactions on Evolutionary Learning*, vol. 3, no. 3, pp. 1–20, 2023.

[32] S. Chang, J. Yang, J. Choi, and N. Kwak, "Genetic-gated networks for deep reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[33] R. Houthooft, Y. Chen, P. Isola, B. Stadie, F. Wolski, O. Jonathan Ho, and P. Abbeel, "Evolved policy gradients," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[34] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, pp. 127–190, 1999.

[35] H. T. Nguyen, K. Tran, and N. H. Luong, "Combining soft-actor critic with cross-entropy method for policy search in continuous control," in *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2022, pp. 1–8.

[36] 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*. PMLR, 2018, pp. 1861–1870.

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

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

[39] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent data analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[40] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches: revisited," *Handbook of metaheuristics*, pp. 453–477, 2019.

[41] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[42] Y. Yang, H. Modares, K. G. Vamvoudakis, and F. L. Lewis, "Cooperative finitely excited learning for dynamical games," *IEEE Transactions on Cybernetics*, vol. 54, no. 2, pp. 797–810, 2023.

[43] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[44] P. Li, Y. Zheng, H. Tang, X. Fu, and J. Hao, "Evorainbow: Combining improvements in evolutionary reinforcement learning for policy search," in *Forty-first International Conference on Machine Learning*, 2024.

[45] Y. Zhang, Z. Mei, X. Wu, H. Jiang, J. Zhang, and W. Gao, "Two-step diffusion policy deep reinforcement learning method for low-carbon multi-energy microgrid energy management," *IEEE Transactions on Smart Grid*, 2024.

[46] E. L. Ratnam, S. R. Weller, C. M. Kellett, and A. T. Murray, "Residential load and rooftop pv generation: an australian distribution network dataset," *International Journal of Sustainable Energy*, vol. 36, no. 8, pp. 787–806, 2017.

[47] C. Miller, A. Kathirgamanathan, B. Picchetti, P. Arjunan, J. Y. Park, Z. Nagy, P. Raftery, B. W. Hobson, Z. Shi, and F. Meggers, "The building data genome project 2, energy meter data from the ashrae great energy predictor iii competition," *Scientific data*, vol. 7, no. 1, p. 368, 2020.