Evolutionary Reinforcement Learning with Late-start Evolution and Clustering Archive

Qiuting Cai, Ya-Hui Jia, *Member, IEEE,* Kaitong Zheng, *Student Member, IEEE,* Shiqi (Shawn) Ou , Wei-Neng Chen, *Senior Member, IEEE*

***Abstract*—Evolutionary Reinforcement Learning (ERL) is a new learning paradigm that integrates Evolutionary Algorithm (EA) with Reinforcement Learning (RL). Existing ERL methods encounter a problem of poor balance between individual quality and diversity, which causes experience mismatch where delayed experiences generated by the population hinder the training of the RL agent. To address this problem, we propose a Late-start Clustering Evolutionary Reinforcement Learning (LCERL) algorithm to improve individual quality and diversity, thereby enhancing the synergy between the population and the RL agent. First, a late-start strategy is proposed to avoid the detrimental impact of poor experiences generated by the population on the RL agent's training in the early stage. Second, a double opposite proximal mutation operator is designed and applied to the RL agent to generate high-quality individuals that are comparable to the RL agent. Third, a clustering selection method with an archive is designed to select diverse individuals for experience generation. Experimental results on the MuJoCo benchmark and a real-world energy management problem demonstrate the superior performance and practicability of LCERL.**

***Index Terms*—Evolutionary Reinforcement Learning, Evolutionary Algorithms, Deep Reinforcement Learning, Late Start Strategy**

I. Introduction

DEEP reinforcement learning (DRL) [1] leverages the decision-making capabilities of reinforcement learning (RL) [2] with the representation power of deep learning [3], enabling agents to solve complex, high-dimensional tasks effectively. It has demonstrated great capabilities in many applications, such as GO [4], games [5], vehicle routing [6], [7], scheduling [8], [9], and robot control tasks [10]–[12]. Despite DRL's remarkable effectiveness, it still faces some fundamental challenges, such as temporal credit assignment with sparse rewards, lack of effective exploration, and brittle convergence properties [13]. To address these issues, Evolutionary Algorithms (EAs) have recently been introduced into DRL [14]–[16]. EAs are gradient-free optimization methods that have great global search ability. When EAs are used in RL environment, they can use the total return of an episode as the fitness indicator, making them indifferent to reward sparsity and robust to environmental noise [17].

Recognizing potential synergistic effects, numerous methods have been developed to integrate EAs with RL, aiming to harness their complementary advantages for more efficient policy search. Khadka and Tumer first proposed the Evolutionary Reinforcement Learning (ERL) paradigm based on the actor-critic algorithm [18]. ERL incorporates an RL agent optimized by gradient-based methods and an actor population evolved by EA. ERL leverages the actor population to provide diverse experiences, which are injected into a replay buffer shared by the population and the RL agent. The RL agent is trained using experiences sampled from the shared replay buffer. Meanwhile, it periodically synchronizes its actor with the worst individual in the population, facilitating the injection of gradient information into the population. The original ERL algorithm employed two basic genetic operators, namely multipoint crossover and Gaussian mutation, to generate new actors. However, they may cause catastrophic forgetting, which is harmful to the evolutionary process. Subsequently, many new genetic operators were proposed, such as distillation crossover and proximal mutation [19], [20], which are more stable and efficient than the operators used in the original ERL. Some works [21], [22] also tried to combine the Evolution Strategy (ES) algorithms like Cross-Entropy Method (CEM) [23] with Twin Delayed Deep Deterministic Policy Gradient (TD3) [24] for better stability.

Despite the superior performance of these new operators, from the perspective of RL, the following three open issues should be solved to achieve harmony between EA and RL. 1) Population initialization is critical for EA. To enhance diversity, random initialization is commonly employed. However, in ERL algorithms, random initialization can induce early-stage experience pollution. The individuals generated by random initialization are diverse but poor in the early stage, which introduces a lot of low-quality experiences into the shared buffer. Given the primacy bias inherent in DRL, which tends to overfit early experiences [25], these low-quality experiences can significantly hinder the RL agent's subsequent learning efficiency and the overall direction of policy optimization. 2) The inherent randomness of EAs inevitably leads to the appearance of poor individuals during evolution. The low-

The work was supported in part by the National Natural Science Foundation of China under Grant 62206098, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515012896, in part by the Introduced Innovative Research and Development Team of Guangdong under Grant 2023ZT10L145, in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076.(*Corresponding authors: Ya-Hui Jia.*)

Qiuting Cai, Ya-Hui Jia, and Shiqi (Shawn) Ou are with School of Future Technology, South China University of Technology, Guangzhou, China. (email: 202321063442@mail.scut.edu.cn; jia.yahui@foxmail.com; sou@scut.edu.cn).

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).

Wei-Neng Chen is with School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. (email: cwnraul634@aliyun.com).

quality experiences generated by these individuals are one reason for the experience mismatch problem that hinders the training of RL [26]. 3) Sometimes, the diversity of the population is poor, and many individuals perform similarly. Under such circumstances, many similar experiences will be generated, which may attract RL into poor local optima.

Considering these three problems, we propose a Late-start Clustering Evolutionary Reinforcement Learning (LCERL) algorithm for more effective policy learning. The novelties of LCERL are as follows:

- **Late-start Strategy:** To mitigate the early-stage experience pollution and primacy bias problems, we propose a late-start strategy that delays the start of the EA process. In contrast to traditional EA methods that rely on random initialization, the late-start strategy ensures that the RL agent first learns high-quality policies before evolutionary strategies are introduced, enhancing learning efficiency in the later stages and final performance.
- **Double Opposite Proximal Mutation:** Aiming at the "experience mismatch" problem, to ensure the generation of high-quality actors, we propose a double opposite proximal mutation operator that directly applies to the RL agent. It collaborates with the stochastic gradient descent mutation operator to help the RL agent explore more search space and ensures that the generated actors match the level of the RL agent.
- **Clustering Selection with Archive:** Aiming at the "local optima" problem, to maintain the diversity of experiences in the shared replay buffer, we propose a clustering selection method with an archive. This method selects actors with widely different behavior characteristics from the archive to achieve diverse experience generation.

The performance of the proposed LCERL algorithm is verified in six MuJoCo benchmark environments and a real-world multi-energy microgrid management problem, compared with the state-of-the-art DRL and ERL algorithms.

The rest of this paper is organized as follows. Section II reviews the related work about ERL. The proposed algorithm is demonstrated in detail in Section III. The performance of LCERL is empirically studied in Section IV. Finally, Section V draws the conclusion.

II. Related Work

The integration of RL and EA shows different forms in different applications. Based on the integration mechanisms and optimization goals, these studies can be divided into three categories: EA-assisted Optimization of RL, RL-assisted Optimization of EA, and Synergistic Optimization of EA and RL [27].

1) EA-assisted Optimization of RL: EA-assisted optimization of RL leverages EAs to enhance various stages of RL training, mitigating problems such as exploration inefficiency, locality of gradient-based optimization, and sensitivity to hyperparameters. A prevalent strategy involves directly evolving the parameters of policy networks or value functions. Representative works, such as Evolved Q-maps (EQ) [28] and Value Function Search (VFS) [29] demonstrate how population-based critic optimization and dual-scale perturbation search can improve policy learning by enhancing value estimation and gradient quality. Beyond parameter optimization, EAs are also employed for dynamic hyperparameter tuning during training. Methods like Population-Based Training (PBT) [30], Online Meta-learning by Parallel Algorithm Competition (OMPAC) [31], and GA-DRL [32] adapt key parameters (e.g., learning rates, discount factors) through online EAs, while Sample-efficient automated deep reinforcement learning (SEARL) [33] further integrates network architecture evolution for automated RL optimization. Additionally, approaches such as Go-Explore [34] and population-guided novelty search-RL (PNS-RL) [35] apply the principles of EA to improve exploration and reward shaping. Overall, this line of research offers enhanced exploration capabilities, robustness to local optima, and adaptive training dynamics, though it often incurs increased computational costs and complexity in evolutionary operator design and hyperparameter management. In a related direction, recent studies have explored hybrid frameworks combining Evolutionary Strategies (ES) with model-based RL. For example, PETS [36] and PlaNet [37] improve model-based RL by integrating probabilistic ensemble models and latent dynamics models with CEM and model predictive control (MPC) to enhance action planning. PETS uses an ensemble model to capture system uncertainties and improve exploration, while PlaNet learns environment dynamics from pixel observations for real-time planning. Grad-CEM [38] combines ES with gradient-based optimization to refine action sequences, boosting CEM's efficiency. These algorithms utilize EAs to choose good actions, diminishing the experience mismatch problem between the estimated model and the real environment. Different from them, the experience mismatch problem discussed in this paper means the mismatch between the experiences generated by the population actors and by the RL actor. It is harmful to the training of the RL actor [26].

2) RL-assisted Optimization of EA: RL-assisted optimization of EAs incorporates RL into the evolutionary process to address challenges such as inefficient population initialization, uncontrolled evolutionary operators, and high evaluation costs. In NGGP [39] and RL-guided GA [40], RL is employed to guide population initialization, enhancing the quality of initial solutions and accelerating convergence. Utilizing RL's strengths in adaptive decision-making and efficient experience use, RL-GA [41] and GSF [42] allow the dynamic selection of crossover and mutation operators based on the current optimization context, thus improving search efficiency and robustness. Furthermore, RL plays a crucial role in quality diversity (QD) optimization by balancing exploration and exploitation to maintain diverse high-performing solutions, which is particularly valuable in multi-objective and constrained problems. Approaches such as QD-RL [43] and PGA-ME [44] highlight RL's effectiveness in guiding diverse solution discovery. While these methods enhance EA adaptability, sample efficiency, and overall performance, they introduce additional hyperparameters and experimental complexity and currently lack strong theoretical guarantees regarding convergence.

3) The Synergistic Optimization of EA and RL: The synergistic Optimization of EA and RL is about how these two techniques collaborate, mutually enhancing each other to reach the

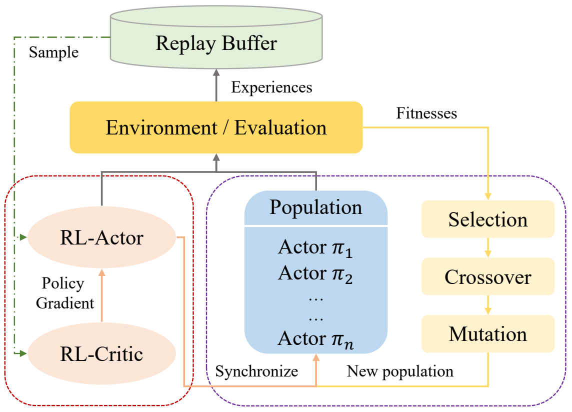


Fig. 1. Overview of the ERL framework, highlighting the synergy between gradient-free evolutionary algorithm (purple box) and gradient-based reinforcement learning (red box) in policy search.

same goal. The ERL paradigm discussed in this paper belongs to this category, which is a hybrid framework that combines the strengths of RL and EA for better policy optimization [18]. The canonical ERL framework is depicted in Fig. 1. The ERL framework begins with initializing an actor-critic RL agent and a population of actor networks, i.e., individuals, with random weights. The fitness of each actor is the cumulative return they achieve for each episode. Subsequently, a selection mechanism based on fitness values selects a subset of individuals. The crossover and mutation operators will be applied to the subset to generate offspring individuals as a new population. During the evaluation of newly generated actors, the experiences obtained from their interactions with the environment are injected into the shared replay buffer. The RL agent samples experiences from the shared replay buffer and updates the parameters of its actor and critic networks via stochastic gradient descent. Additionally, the weights of the RL actor are periodically copied into the worst-performing individual in the population via synchronization. The synchronization allows EA to leverage the knowledge learned through gradient descent directly. The shared replay buffer and synchronization facilitate the flow of information between the EA and RL components, thereby enhancing the synergy between them.

After the first ERL framework [18], researchers have proposed various ERL algorithms to exploit the synergy. These algorithms improve the synergy from two aspects: 1) incorporating new evolution operators or evolutionary computation algorithms and 2) designing new integration methods or frameworks.

Evolutionary computation algorithms play a crucial role in ERL. Some researchers focused on replacing the basic genetic operators of Genetic Algorithm (GA) [45] with new operators to improve the algorithms' efficiency and stability. Proximal Distilled Evolutionary Reinforcement Learning (PDERL) [19] uses learning-based mutation operators to replace pure random mutation to achieve better training efficiency. Meanwhile, researchers found that Cross-Entropy Method (CEM) can explore the search space by adjusting the probability distribution used to create samples, without directly using crossover and mutation operations [46], [47]. Its evolutionary mechanism can automatically exclude unstable regions of policy updates. Building on this foundation, Pourchot *et al.* proposed Cross-entropy Method-Reinforcement Learning (CEM-RL) [21], combining CEM with Deep Deterministic Policy Gradient (DDPG) [48] and TD3. Tang *et al.* proposed Cross-entropy Method-Actor-critic with Experience Replay (CEM-ACER) [22], which combines CEM with the off-policy actor-critic algorithms with experience replay, leveraging the strengths of both evolutionary strategies and off-policy updates. The CEM-RL framework has served as the basis for the development of several algorithms, such as NERL [49] and CEM-SAC [50]. Particle Swarm Optimization (PSO) [51] is also adopted in ERL for different purposes. Evolutionary Action Selection-Twin Delayed Deep Deterministic Policy Gradient (EAS-TD3) [52] uses PSO to optimize actions rather than policy parameters. Zhu *et al.* proposed a Two-stage Evolutionary Reinforcement Learning (TERL) [53] algorithm, which divided the learning process into exploration and exploitation phases, using PSO and shared replay buffers to facilitate information exchange among individuals.

Besides introducing new evolution operators and algorithms, some other researchers proposed new integration methods or ERL frameworks. Khadka proposed Collaborative Evolutionary Reinforcement Learning (CERL) [54] to mitigate the hyperparameter sensitivity and poor exploration in RL algorithms. CERL combines multiple RL learners, each with distinct time horizons, with EA to explore diverse regions of the solution space, thereby improving sample efficiency. To address overestimation bias and inefficient mutation issues, Federico *et al.* proposed X-DDPG [55], which combines GAs with DRL and injects more robust individuals into the population. Evolutionary Reinforcement Learning algorithm enhanced with Truncated variance and Distillation mutation (ERL-TD) [20] utilizes multiple networks to evaluate state-action pairs, providing more accurate evaluation results. Enrico *et al.* [56] proposed a new framework called Soft Updates for Policy Evolution (Supe-RL), which periodically perturbs the RL agent's actor parameters to create a population and updates the actor's parameters through soft updates if the best individual outperforms the RL agent. Suri *et al.* [57] introduced Evolution-based Soft Actor-Critic (ESAC) that integrates ES with Soft Actor-Critic (SAC) to solve high-dimensional continuous control tasks, leveraging techniques like automatic mutation tuning, soft winner selections, and hindsight crossovers. To address the challenges of low sample efficiency, learning instability, and insufficient exploration in sparse-reward environments of DRL, Zheng *et al.* proposed Cooperative Heterogeneous Deep Reinforcement Learning (CHDRL) [58]. CHDRL integrates off-policy, on-policy, and evolutionary agents to enhance exploration and learning efficiency through a cooperative mechanism. Wu *et al.* proposed an Adaptive Evolutionary Reinforcement Learning (AERL) algorithm [59], introducing an adaptive mutation operator and an early termination strategy to enhance efficiency and performance in continuous control tasks. To solve the problem of high computational cost and low sample efficiency caused by fitness evaluation relying heavily on real environment interactions, Wang *et al.* proposed the surrogate-assisted controller

TABLE I
A COMPARISON OF SOME REPRESENTATIVE ERL ALGORITHMS.

Algorithm	Operators	RL Algorithm	Experience Diversity Control	RL and EA Synergy
ERL [18]	multi-point crossover, Gaussian mutation	DDPG	No	Synchronous parallel
PDERL [19]	distillation crossover, proximal mutation	DDPG	No	Synchronous parallel
CEM-RL [21]	Cross-Entropy Method	TD3	No	Synchronous parallel
NERL [49]	Cross-Entropy Method	TD3	novelty search	Synchronous parallel
SUPE-RL [56]	Gaussian mutation	Rainbow/PPO	No	Synchronous parallel
ERL-Re^2 [61]	multi-point crossover, Gaussian mutation	TD3 with shared network	No	Synchronous parallel
ERL-TD [20]	distillation mutation	SAC with truncated variance	No	Synchronous parallel
CoERL [62]	Gaussian mutation	SAC	No	Synchronous parallel
TR-ERL [26]	canonical evolution strategy	TD3	No	Synchronous parallel
TERL [53]	PSO-based population update	TD3	No	Synchronous parallel
LCERL(ours)	double opposite proximal mutation SGD-based mutation	TD3	Clustering Selection with Archive	Late start Strategy

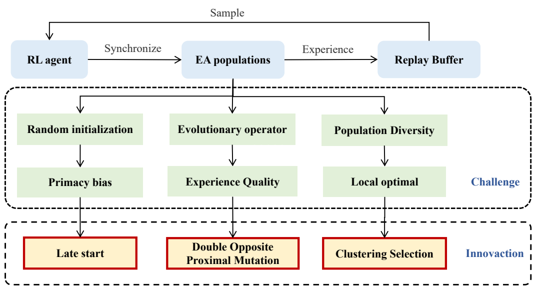


Fig. 2. The motivation and design rationale of LCERL.

[60] for evolutionary reinforcement learning frameworks. This framework uses a surrogate model to approximate fitness evaluations, reducing computational costs while maintaining performance. Evolutionary Reinforcement Learning with a Two-scale State (ERL-Re^2) [61] allows EA and RL strategies to share the same nonlinear state representation while maintaining separate linear policy representations. To illustrate the difference between our algorithm and some representative related works clearly, we use a table to comprehensively compare representative ERL algorithms in operators, RL algorithm, experience diversity control, and the synergy between RL and EA, as shown in Table I. More detailed comparisons can be found in [27].

Despite the development of numerous complex frameworks and operators, the problem of poor balance between the quality and diversity of population experiences has not been settled well.

III. Proposed Algorithm

This section first introduces the overall structure of LCERL, followed by a detailed explanation of its three key components: the late-start strategy, double opposite proximal mutation, and clustering selection with archive.

A. Structure of LCERL

LCERL is proposed to train a good RL actor efficiently. The primary objective of the population in LCERL should be emphasized that it is used to help RL training by generating more high-quality and diverse experiences. For a more intuitive comprehension of LCERL, its motivation and design rationale are shown in Fig. 2. In evolutionary reinforcement learning, population initialization, evolutionary operator, and population diversity are crucial; if not handled well, they can lead to issues like primacy bias, degradation of population quality, and getting trapped in local optima. To address these issues, We introduce a late-start strategy, ensuring high-quality early experiences, double opposite proximal mutation to align generated actors with the RL agent's learning phase, and clustering selection with archive to maintain diversity and

Algorithm 1 LCERL

```
Input: RL actor $\pi$, start criterion of EA $T_{EA}$, replay buffer $D$, archive $A$,
population size $N$, archive size $AZ$, cluster number $K$
Output: best actor $\pi*$
 1: Randomly initialize $\pi$
 2: $D = \emptyset$, $A = \emptyset$, $j = 0$
 3: while the stop criterion is not met do
 4:     while interaction does not finish do
 5:         $\pi$ interacts with the environment
 6:         Store transition $(s, a, s', r)$ in $D$
 7:         Sample experiences from $D$ and train $\pi$
 8:         $j \leftarrow j + 1$
 9:     end while
10:     if $j \geq T_{EA}$ then
11:         Copy $\pi$ $N$ times to form a new population $pop$
12:         for each $\pi_i$ in $pop$ do
13:             if $i\%2 == 0$ then
14:                 Apply double opposite proximal mutation
15:             else
16:                 Apply stochastic gradient descent
17:             end if
18:             $f(\pi_i) \leftarrow$ Evaluate $(\pi_i)$
19:             Add $(\pi_i, f(\pi_i))$ to $A$
20:             if $|A| > AZ$ then
21:                 Remove the worst actor from $A$
22:             end if
23:         end for
24:         $E_K \leftarrow$ Clustering_Selection$(A, D, K)$
25:         $E_M \leftarrow$ Select top $M = N - K$ actors from $pop$
26:         $E \leftarrow E_K \cup E_M$
27:         for $\pi_i$ in $E$ do
28:             while interaction does not finish do
29:                 $\pi_i$ interacts with the environment
30:                 Store transition $(s, a, s', r)$ in $D$
31:                 Sample experiences from $D$ and train $\pi$
32:                 $j \leftarrow j + 1$
33:             end while
34:         end for
35:     end if
36: end while
37: $\hat{\pi} \leftarrow$ best actor in $E$
38: return $\pi* = $ argmax$(f(\pi), f(\hat{\pi}))$
```

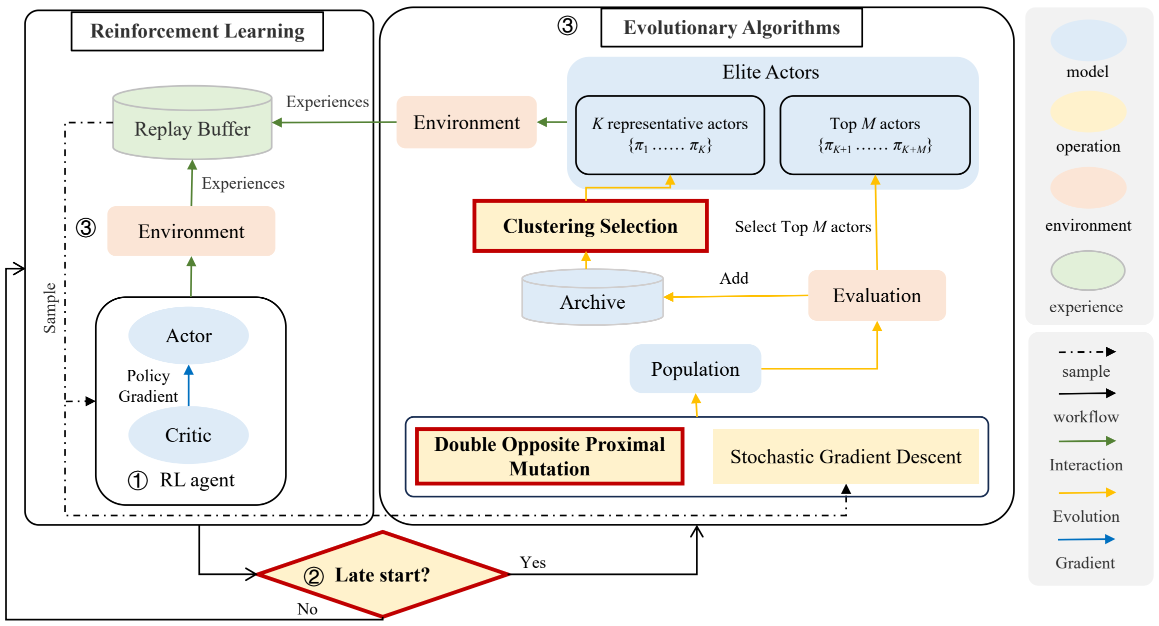


Fig. 3. Framework of LCERL. The left part is the RL part, and the right part is the EA part, where the three red boxes are the innovation points proposed in this paper.

improve exploration, ultimately enhancing learning efficiency and stability. The framework of LCERL is illustrated in Fig. 3. Specifically, LCERL integrates EA into the TD3 algorithm, which comprises an interactive environment, a shared replay buffer, an RL agent, and an archive that stores recent actors mutated from the RL actor. Comparing Fig. 3 and Fig. 1, we can find that LCERL does not explicitly maintain a population. New individuals are always generated by mutating the RL actor. This practice is beneficial to alleviating the experience mismatch problem where the experience generated by the population does not match the level of the RL agent [26]. Based on this framework, the late-start strategy, double opposite proximal mutation, and clustering selection strategy are embedded, all aiming to generate diverse and high-quality experiences for the RL training.

The training process of LCERL is shown in Algorithm 1. The first step is to initialize these components (lines 1-2). We use j to count how many timesteps the RL agent has been trained. After initialization, the main loop begins. In each iteration, the RL actor will first interact with the environment and sample experiences from the shared buffer to update itself and the critic (lines 4-9). In the early stage, i.e. when $j < T_{EA}$, the late start strategy will skip the EA part. Only the RL actor interacts with the environment, allowing the RL to focus on training based on its own experiences. The EA component is activated when a predefined condition is met (line 10), i.e. $j \geq T_{EA}$. In the EA part, two mutation operators will be applied to the RL actor to generate a population of actors and these new actors will be added to the archive. The archive in LCERL is set with a fixed capacity to store the most recent and high-performing actors generated by the mutation operators and their fitness f_i, i.e., cumulative reward values. The reason for setting a fixed capacity is to maintain the novelty and quality of actors in the archive and to decrease the complexity of the clustering selection process. Based on the current RL actor, we generate a new population of actors using two mutation methods: double opposite proximal mutation and stochastic gradient descent (lines 11-17). Then, these new actors are inserted into the archive. If the archive exceeds its maximum size, it will discard the worst actor based on fitness (lines 18-22). Next, an elite actor set E is selected to provide experiences. It consists of two parts. K elite actors come from the archive where the cluster selection method is applied, denoted as E_K (line 24). The clustering selection function is demonstrated in Algorithm 2. The other elite actors come from the newly generated population where the top $M = N - K$ actors are considered, denoted as E_M (line 25). N is the total number of elite actors. For the sake of simplicity, the number of elite actors is set equal to the size of the population. Then, these elite actors will interact with the environment to generate high-quality and diverse experiences, which are added to the shared replay buffer (lines 27-34) to train the RL agent. Finally, the best actor in the final population is compared with the RL actor. The better one is returned as the final model (lines 37-38).

B. Late-start strategy

During the early stage, the strong exploration of EA tends to introduce excessive randomness, which may generate a lot of low-quality actors and low-reward experiences. Since DRL uses neural networks instead of discrete tables, the great efforts to approximate the low-reward actions and states in the early stage have no benefits and will seriously influence the subsequent optimization. Although in LCERL, all EA actors are generated by mutating the RL actor, since RL performs poorly in the early stages, the population generated from these mutations provides limited benefits.

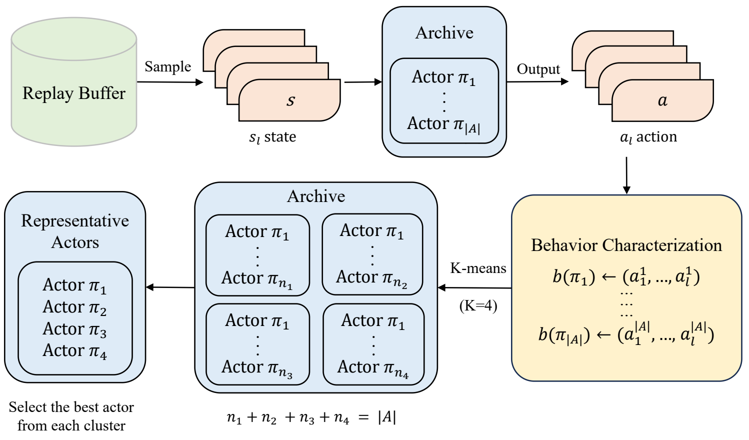


Fig. 4. The process of clustering selection. Assume there are $K = 4$ clusters.

To mitigate this issue, we propose the late-start strategy. The core idea behind this strategy is to delay the introduction of intensive exploration, allowing the RL agent to stabilize its learning trajectory before further increasing the exploration. This strategy enables RL to effectively use gradient information to learn the basic policy in the early stage, avoiding interference caused by EA randomness.

Since the late-start strategy is crucial to the RL agent training, in LCERL, we use the number of RL training timesteps as the measurement to start the EA component. For the sake of simplicity, assuming the total number of RL training timesteps is T_{ep}, the starting criterion for EA denoted as T_{EA} is defined as:

$$T_{EA} = \delta \cdot T_{ep} \tag{1}$$

where δ is a hyper-parameter.

C. Double Opposite Proximal Mutation

The double opposite proximal mutation is designed based on the proximal mutation [19]. The proximal mutation is an operator that uses sensitivity y to scale the Gaussian perturbation applied to each weight of the actor network π_θ through

$$\theta \leftarrow \theta + \frac{x}{y}, \tag{2}$$

with $(\mathbf{x} \sim \mathbf{N}(\mathbf{0}, \sigma\mathbf{I}))$, where σ is a mutation magnitude hyperparameter and the initial value is 1. The sensitivity y is calculated by the gradient of each dimension of the output action over N_{state} states, which are sampled from the buffer of actors. It is defined as:

$$y = \sqrt{\sum_{p=0}^{D_{action}} \left(\sum_{q=0}^{N_{state}} \nabla_\theta \pi_\theta(s_q)_p \right)^2} \tag{3}$$

Each actor maintains a personal buffer to store recent experiences in addition to the shared replay buffer. D_{action} is the dimension of the action space and N_{state} denotes the number of sample states s_q.

To augment the mutation efficacy, we introduce an opposite proximal mutation. It introduces a scaling factor α and compares the result of the opposite proximal mutation with the proximal mutation, ultimately adopting the better one, following as:

$$\theta \leftarrow \theta - \alpha * \frac{x}{y} \tag{4}$$

$$\alpha = \left| \frac{\text{eval}_{\text{after}}}{\text{eval}_{\text{before}}} \right| \tag{5}$$

The values $\text{eval}_{\text{before}}$ and $\text{eval}_{\text{after}}$ respectively represent the evaluations before and after the proximal mutation. When the evaluation improves, $\alpha > 1$ signals the need for a larger mutation to explore further. When the evaluation worsens, $\alpha < 1$ suggests reducing the mutation amplitude to maintain the current solution. To enhance mutation effectiveness, we introduced a re-mutation threshold. If the mutation fails to improve performance, the mutation standard deviation σ is increased by a factor of 1.5, as shown by:

$$\sigma' = 1.5 \cdot \sigma \tag{6}$$

Based on the new σ', perform a new round of opposite proximal mutation. The double opposite proximal mutation uses two opposite perturbations to enable broader exploration and refine the search around the current solution, avoiding local optima.

D. Clustering Selection with Archive

In EA, individuals are typically characterized by both their genotype and phenotype. The genotype refers to the underlying genetic representation. The phenotype pertains to the observable traits, like the actions exhibited by an individual in response to the environment. In LCERL, we specifically use the phenotype to describe each actor, i.e. the actions it takes. This is because the actions taken by the actor in response to a given state, rather than its internal genetic structure, are directly relevant to its performance in the environment and the experiences it generates. To efficiently select diverse elite actors from the archive, a clustering selection method is designed.

The detailed process is shown in Fig. 4 and Algorithm 2. Initially, we sample l states randomly from the shared

Algorithm 2 Clustering_Selection

```
Input: archive $A$, replay buffer $D$, cluster number $K$
Output: archive elite actors $E_K$
1: Randomly select $l$ states $\{s_1, \dots, s_l\}$ from $D$
2: for each actor $\pi_i$ in $A$ do
3:     Use $\{s_1, \dots, s_l\}$ to obtain actions $\{a_1, \dots, a_l\}$
4:     Behavior Characterization $b(\pi_i) \leftarrow \{a_1, \dots, a_l\}$
5: end for
6: Cluster actors in $A$ using K-means based on $\{b(\pi_1), \dots, b(\pi_{|A|})\}$
7: $E_K \leftarrow$ Select the best actor from each cluster
8: return $E_K$
```

replay buffer and obtain the actions $\{a_1^i, \dots, a_l^i\}$ of each actor π_i on these l states (lines 1-5). These actions are then treated as the behavioral characteristics of the actors, denoted as $b(\pi_i)$, as they reflect how each actor interacts with its environment. Then, the K-means algorithm is executed based on the behavioral characteristics $\{b(\pi_1), \dots, b(\pi_{|A|})\}$ to divide the actors into K clusters, where each cluster represents a set of actors with similar behaviors in response to the sampled states (line 6). Finally, we select the best actor from each cluster, as determined by their fitness (line 7). Through this clustering selection method, we can choose K representative actors in the archive to generate diverse experiences.

E. Discussions

Here, we discuss the rationality of three innovations in LCERL.

1) Late-start strategy: Given an episode $\{s_0, a_0, r_0, s_1, a_1, r_1, \dots\}$, the actor-critic algorithm will update its critic network V_ω and actor network π_θ following:

$$\epsilon_t = r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t) \tag{7}$$

$$\omega = \omega + \alpha_\omega \sum_t \epsilon_t \nabla_\omega V_\omega(s_t) \tag{8}$$

$$\theta = \theta + \alpha_\theta \sum_t \epsilon_t \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{9}$$

where $V_\omega(s) = \sum_a \pi(a|s) Q^\pi(s,a)$, r_t is the immediate reward, γ is the discount factor, α_ω and α_θ are two learning rate hyper-parameters.

The early-stage injection of large amounts of random low-quality experience from the population causes the critic to prematurely and incorrectly fit underestimated Q values, i.e. $Q^\pi(s,a)$, which may lead to two situations. The first situation is that V_ω can tell which state is better but due to the overabundance of low-return episodes, the estimated values for different states tend to be similar. In such a case, ϵ_t will be relatively small and close to r_t, making the algorithm short-sighted and decelerating the update of the critic parameters ω and the actor parameters θ. The second situation is that V_ω cannot judge which state is better. Under such circumstances, ϵ_t would give wrong update directions. As a consequence, the agent tends to spend an extended amount of time exploring low-quality regions, which negatively impacts the efficiency of exploration in reinforcement learning.

These analyses highlight that low-quality early experience from EA actors undermines the RL training stability and learning efficiency, potentially leading to suboptimal convergence. The late-start strategy mitigates this by delaying EA involvement, thus improving overall performance.

2) Double Opposite Proximal Mutation:

The double opposite proximal mutation explores both positive and negative directions, and there is a certain probability of re-exploration, enhancing search space coverage. This approach not only improves local optimization but also broadens the search area, facilitating escape from local optima. Moreover, the mutation operator has a greater probability of addressing the issue of saddle points, common in high-dimensional optimization problems. This mutation operator increases the chance of escaping flat or deceptive regions, thereby improving the chances of finding better solutions.

However, since the proposed double opposite proximal mutation strategy may undergo secondary mutation with a certain probability, it leads to additional time consumption. In the experimental section, we will provide a comparative analysis of training time with other algorithms to show that the extra time cost is not significant.

3) Clustering Selection with Archive: Numerous studies have demonstrated that population diversity control strategies significantly impact the quality of the final solution [63], [64]. Although our algorithm does not maintain a population for evolution, the experience generated from the interaction between the actors obtained through mutation and the environment directly contributes to training the RL agent. In some continuous high-dimensional environments, the Q-value function is multi-modal, meaning there are several different strategies that can yield high returns. In LCERL, we use phenotypes to cluster actors within the archive, selecting those with behavioral characteristic differences and high quality to interact with the environment for RL training. This approach helps prevent the RL agent from converging to a local optimum while neglecting other promising local optima. Ultimately, this strategy ensures a balanced exploration-exploitation trade-off, enhancing the algorithm's overall capability.

IV. Experimental Studies

To evaluate the practical effectiveness of the proposed algorithm, we compare it with several well-known ERL and RL algorithms, including ERL [18][1], CERL [54][2], PDERL [19][3], CEM-RL [21][4], ERL-Re^2 [61][5], ERL-TD [20][6], CoERL [62][7], TERL [53], TD3 [24][8], SAC [65], PPO [66], and DDPG [48][9]. These algorithms were selected as benchmarks

[1]https://github.com/ShawK91/Evolutionary-Reinforcement-Learning
[2]https://github.com/intel/cerl
[3]https://github.com/crisbodnar/pderl
[4]https://github.com/apourchot/CEM-RL
[5]https://github.com/yeshenpy/ERL-Re2
[6]https://github.com/2019cyf/ERL-TD
[7]https://github.com/HcPlu/CoERL
[8]https://github.com/sfujim/TD3
[9]https://github.com/openai/baselines/

due to their proven success in various reinforcement learning tasks, providing a solid foundation for evaluating the strengths and weaknesses of our proposed method. For each of these algorithms, we set the hyper-parameters for each algorithm to the recommended values specified in the respective papers or to the default values provided in their original programs. To assess their performance, we apply each algorithm to six complex MuJoCo environments (Ant-v4, HalfCheetah-v4, Hopper-v4, Humaniod-v4, Swimmer-v4, Walker2d-v4) [67] in OpenAI Gym [68]. These environments represent a range of complex, high-dimensional control tasks that require agents to exhibit high levels of decision-making and adaptability.

In addition to this comparison, we also conduct ablation studies to evaluate the individual contributions of the three proposed strategies. These ablation experiments are designed to demonstrate the effectiveness of each strategy in isolation, proving that every component plays a crucial role in enhancing the algorithm's performance. Through these experiments, we aim to not only validate the superiority of our proposed method over existing approaches but also to highlight the specific benefits brought by each of the strategies included in LCERL.

Finally, to show the practicability of LCERL, we also apply it to a real-world multi-energy microgrid (MEMG) energy scheduling problem and compare its performance with the state-of-the-art RL algorithms that were commonly used in the MEMG environment [69].

A. Experiment Setup

For reproducibility, each algorithm is trained on each task with five different random seeds, and the average performance across these five runs represents the algorithm's performance on that task. Following the settings in CEM-RL [21], ERL-Re^2 [61], ERL-TD [20], each training consists of one million training steps ($T_{ep} = 1,000,000$) of the RL agent training. The hyper-parameters used in our method are detailed and corresponding references as follows Table II. The experiments of tuning δ and K are provided in the supplementary material. Finally, the RL adopted in LCERL is TD3. The settings of TD3 are shown in Table III.

TABLE II
PARAMETER SETTING

Parameter	Value	Ref
The parameter of TD3	default setting	[24]
The training timesteps	1,000,000	[21], [61], [20]
The size of population buffer and RL_{agent} buffer	50,000	[26]
The size of individual buffer in population	8,000	[19]
The size of population	10	[18], [19]
The Mutation magnitude parameter σ	0.01	[18]
The batch size N_M	256	[18]
The Archive size AZ	20	[70]
The number of states l for clustering selection	1024	[70]
The parameter of late-start strategy starting criterion δ	0.2	Ours
The parameter of K-means cluster number K	4	Ours

B. Experimental Results

Fig. 5 shows the performance of all algorithms. The mean return values and standard deviations are represented by the solid lines and shaded areas in the figure, respectively. Additionally, we present the final performance of various

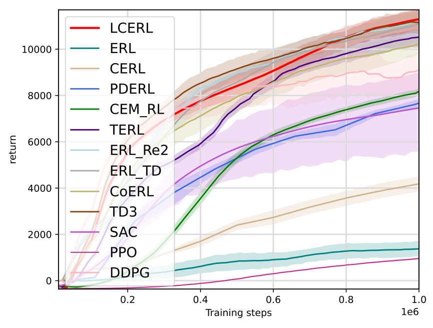
TABLE III
TD3 HYPERPARAMETERS SETTING

Parameter	Value
Actor network	FC(256,256)
Actor activate function	ReLU
Critic network	FC(256,256)
Critic activate function	ReLU
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor	0.99
Target network update rate	0.05
Std of Gaussian exploration noise	0.1
Noise added to target policy	0.2
Range to clip target policy noise	0.5
Frequency of delayed policy updates	2
Batch size for both actor and critic	256

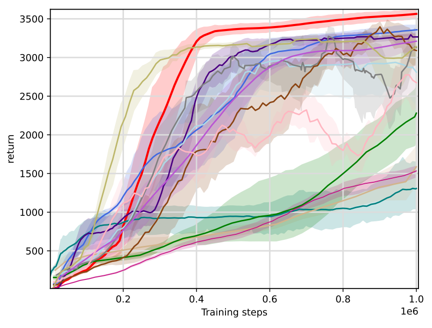
algorithms across six MuJoCo environments in Table IV. Besides the performance statistics, i.e., mean value ± standard deviation, we have also performed the Wilcoxon rank-sum test between LCERL and the compared algorithms in each environment with a significance level set to 0.05. $\uparrow/\downarrow/\updownarrow$ indicates LCERL is significantly better than/worse than/equal to the compared algorithm in Table IV. To get the overall rank of the algorithms and to show whether LCERL is significantly better in general, we have also performed the Friedman test with the Finner post-hoc analysis over all experimental results in all environments, which is shown as the last column.

According to mean values, LCERL outperforms the other algorithms in most environments except Walker2d-v4 and Ant-v4. In Walker2d-v4, our algorithm is slightly inferior to ERL-Re^2 but still ranks second among all algorithms. In Ant-v4, our algorithm is slightly inferior to ERL-TD and ERL-Re^2 and ranks third among all algorithms. ERL-TD utilizes multiple Q-networks to evaluate state-action pairs, allowing for more accurate assessments from the ensemble of networks [20]. ERL-Re^2 combines a shared state representation enriched by value function maximization with individual policy representations optimized through behavior-level genetic operators and a novel policy fitness surrogate [61]. These may be the reason for their good performance in Ant-v4 and Walker2d-v4, since these two environments require higher sample efficiency for value function approximation [19], [53]. Previous researchers think the Swimmer environment has a deceptive gradient phenomenon [71]. The deceptive gradients mean the search space including poor local optima. The gradient-based algorithms are ineffective in the Swimmer. We also can see in Fig. 5 that in Swimmer, the RL algorithms SAC, PPO, DDPG, and TD3 perform poorly, worse than LCERL. LCERL utilizes mutations to prevent RL from getting stuck in deceptive local optima. In Walker2d-v4, Hopper-v4, and Humaniod-v4, some algorithms converge faster initially than LCERL, but their convergence speed gradually weakens. In contrast, LCERL demonstrates more efficient synergy between EA and RL in the later stages. It remains stable and surpasses other algorithms in the mid-to-late stages.

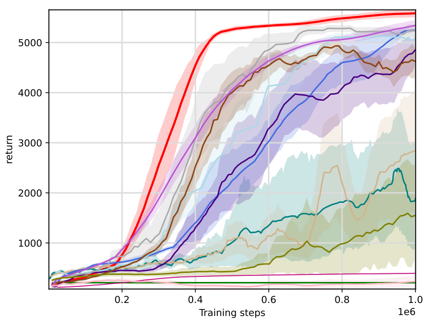
Meanwhile, the Wilcoxon rank-sum test shows that LCERL outperforms other algorithms in most environments. The Friedman test, with a p-value, further emphasizes LCERL's



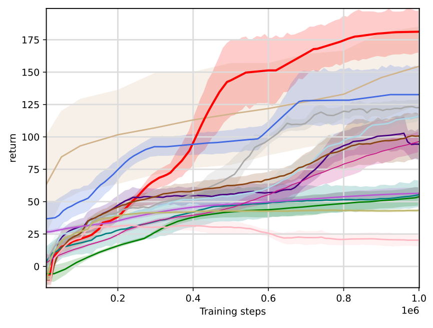
(a) HalfCheetah-v4



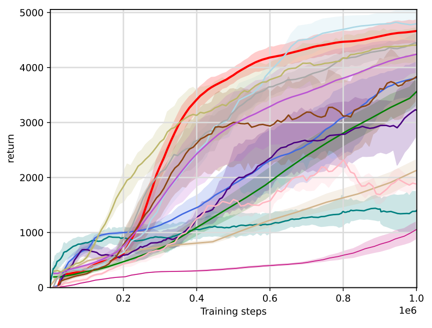
(b) Hopper-v4



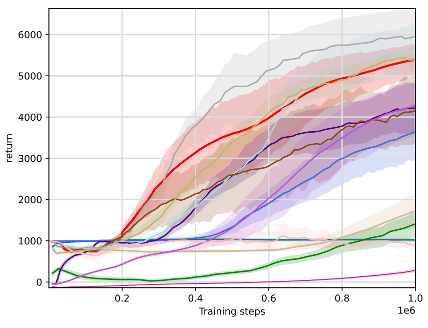
(c) Humanoid-v4



(d) Swimmer-v4



(e) Walker2d-v4



(f) Ant-v4

Fig. 5. The mean return obtained on HalfCheetah-v4 (a), Hopper-v4 (b), Humanoid-v4 (c), Swimmer-v4 (d), Walker2d-v4 (e), Ant-v4(f).

TABLE IV
FINAL PERFORMANCE (MEAN±STD.) ON MUJOCO ENVIRONMENTS WITH STATISTICAL SIGNIFICANCE INDICATORS AND AVERAGE RANK OF ALGORITHMS.

Classification	Algorithm	HalfCheetah-v4	Hopper-v4	Humanoid-v4	Swimmer-v4	Walker2d-v4	Ant-4	Avg. Rank(p-value)
RL	TD3	$10820 \pm 1646(\uparrow)$	$3376 \pm 228(\uparrow)$	$4656 \pm 706(\uparrow)$	$102 \pm 29(\uparrow)$	$4375 \pm 258(\uparrow)$	$4341 \pm 1123(\uparrow)$	5.2(0.003)
	SAC	$7633 \pm 2796(\uparrow)$	$3241 \pm 166(\uparrow)$	$5356 \pm 236(\uparrow)$	$57 \pm 8(\uparrow)$	$4317 \pm 296(\uparrow)$	$4342 \pm 1025(\uparrow)$	6.4(0.0)
	PPO	$1061 \pm 37(\uparrow)$	$1785 \pm 153(\uparrow)$	$405 \pm 12(\uparrow)$	$100 \pm 11(\uparrow)$	$1374 \pm 345(\uparrow)$	$414 \pm 61(\uparrow)$	11.1(0.0)
	DDPG	$9045 \pm 1294(\uparrow)$	$2397 \pm 613(\uparrow)$	$248 \pm 178(\uparrow)$	$20 \pm 8(\uparrow)$	$1885 \pm 753(\uparrow)$	$578 \pm 537(\uparrow)$	10.8(0.0)
ERL	ERL	$1450 \pm 494(\uparrow)$	$1258 \pm 269(\uparrow)$	$2242 \pm 1873(\uparrow)$	$59 \pm 18(\uparrow)$	$1544 \pm 492(\uparrow)$	$1019 \pm 50(\uparrow)$	11.2(0.0)
	CERL	$6748 \pm 725(\uparrow)$	$2735 \pm 14(\uparrow)$	$2862 \pm 1972(\uparrow)$	$170 \pm 38(\updownarrow)$	$3365 \pm 258(\uparrow)$	$2943 \pm 1055(\uparrow)$	7.9(0.0)
	PDERL	$8134 \pm 141(\uparrow)$	$3363 \pm 98(\uparrow)$	$5250 \pm 43(\uparrow)$	$133 \pm 32(\uparrow)$	$3877 \pm 485(\uparrow)$	$3764 \pm 1028(\uparrow)$	6.0(0.0)
	CEM-RL	$8485 \pm 413(\uparrow)$	$2513 \pm 677(\uparrow)$	$207 \pm 1(\uparrow)$	$58 \pm 13(\uparrow)$	$3891 \pm 601(\updownarrow)$	$1813 \pm 600(\uparrow)$	9.2(0.0)
	ERL-Re^2	$10957 \pm 372(\uparrow)$	$2524 \pm 727(\uparrow)$	$4917 \pm 430(\uparrow)$	$120 \pm 10(\uparrow)$	$\mathbf{4935 \pm 250}(\updownarrow)$	$5796 \pm 923(\updownarrow)$	4.4(0.031)
	ERL-TD	$11204 \pm 1060(\updownarrow)$	$2982 \pm 848(\updownarrow)$	$5303 \pm 80(\uparrow)$	$116 \pm 36(\uparrow)$	$4223 \pm 614(\updownarrow)$	$\mathbf{6209 \pm 896}(\updownarrow)$	4.1(0.055)
	CoERL	$10303 \pm 391(\uparrow)$	$3295 \pm 128(\uparrow)$	$2218 \pm 2100(\uparrow)$	$44 \pm 2(\uparrow)$	$4424 \pm 508(\updownarrow)$	$5303 \pm 267(\updownarrow)$	6.3(0.0)
	TERL	$10669 \pm 587(\uparrow)$	$3290 \pm 73(\uparrow)$	$5223 \pm 196(\uparrow)$	$106 \pm 23(\uparrow)$	$3113 \pm 543(\uparrow)$	$4119 \pm 656(\uparrow)$	6.1(0,0)
	LCERL	$\mathbf{11393 \pm 698}$	$\mathbf{3610 \pm 105}$	$\mathbf{5608 \pm 100}$	$\mathbf{181 \pm 27}$	$4712 \pm 319(\uparrow)$	5459 ± 583	**2.2**

$\uparrow/\downarrow/\updownarrow$ indicates LCERL is significantly better than/worse than/equal to the compared algorithm according to the Wilcoxon rank-sum test. The last column shows the average ranks with p-values made by the Friedman test with Finner post-hoc analysis.

superiority by showing that it consistently ranks the highest (average rank of 2.2) across all environments, indicating its overall effectiveness and consistency in diverse tasks. At the standard significance level of 0.05, LCERL significantly outperforms most algorithms, with the only exception being ERL-TD. When the significance level is relaxed to 0.1, LCERL also significantly outperforms ERL-TD, achieving statistically significant improvements over all other algorithms. In conclusion, the results demonstrate that LCERL outperforms other algorithms across multiple tasks, with significant improvements in performance and stability, validating its effectiveness.

C. Ablation Experimental Results

1) Late start strategy: To examine the effectiveness of the late start strategy, we conduct an ablation experiment by removing the late start strategy, allowing EA to take effect from the beginning. Meanwhile, to show that exploration is still important in the late stage of ERL, we also create an early stop strategy for comparison, which stops the EA component early in the late stage. Fig. 6 shows the learning curves.

Comparing LCERL with and without the late start strategy, we can find an interesting phenomenon in each environment that the full LCERL version initially lags behind but gradually catches up and finally overtakes the one without the late start strategy. This is because LCERL initially relied solely on an RL actor, which the exploration ability of LCERL before 200,000 timesteps worse than without using the late start strategy. The probability of initializing ten population actors with a good one is inherently higher than initializing just one actor. Although initially applying EA can temporarily enhance algorithm performance, an overabundance of low-return experiences injected into the shared replay buffer will hinder the learning efficiency of RL, ultimately limiting performance improvements in the later stages. In contrast, the late start strategy makes RL establish the correct optimization direction during the early stage and then utilizes EA to enhance exploitation capabilities. This strategy not only mitigated the negative effects of early randomness but also optimized the synergy between RL and EA. Despite the initial decline in performance, the overall performance improvement and increased training stability demonstrate the effectiveness of the late start strategy.

Comparing the blue and black curves, we can find that the EA component, especially mutation, still plays a vital role in maintaining convergence stability and efficiency in the later stage of ERL. Stopping EA in the later stage is a wrong decision. This is because, in the later stage, the RL agent has already approximated the values of most states and actions. Keeping exploring more states is beneficial to the accuracy of the model. Instead, if only the RL agent itself explores the environment, the experiences in the buffer will become similar, which will make the RL agent overfit to the states that it used to visit. Overall, the experiment shows that the late start strategy is very useful.

2) Double Opposite Proximal Mutation: To examine the effectiveness of the double opposite proximal mutation strategy, we design a comparative experiment that replaces the double opposite proximal mutation with the traditional proximal mutation method. The results are shown in Fig. 7. The experimental results demonstrate that the double opposite proximal mutation method outperforms the traditional proximal mutation in most environments. Notably, in the Hopper-v4, Humanoid-v4, Swimmer-v4, and Walker2d-v4, our method proves beneficial throughout nearly the entire process. These findings highlight that the double opposite proximal mutation strategy significantly contributes to the generation of higher-quality experiences, which enhances the efficiency and effectiveness of RL training.

3) Clustering Selection with Archive: To examine the effectiveness of the clustering selection with an archive, we conduct an ablation experiment by removing the clustering selection and the archive, relying solely on mutation methods for population generation and experience generation. Fig. 8 illustrates the learning curves with and without the clustering selection. The experimental results show that in most environments, the clustering selection method with the archive outperformed the approach without it. Although on two environments, HalfCheetah-v4 and Hopper-v4, it does not exhibit significant benefits, there are no adverse effects observed either. Meanwhile, our method consistently benefits RL training throughout the process in other environments. Therefore, it demonstrates that the proposed clustering selection method effectively promotes the generation of diverse experiences.

D. Further Analysis

From the above results, we observed that the performance gap of the ablation experiment is more obvious in the Walker2d-v4 environment than in other environments. Meanwhile, most algorithms have stable convergence behavior in this environment. Thus, we select the Walker2d-v4 environment to make a deep analysis about LCERL.

1) Late-start Strategy: To empirically support our theoretical analysis in Section III.E, we examine two core metrics that reflect different aspects of actor-critic learning: Q value (value estimation) and critic loss (value stability), with and without the late-start strategy as shown in Fig. 9. The Q value is calculated as the average Q value of a batch of sampled experiences from the buffer.

According to the Bellman optimality equation, we know that a policy π^* is optimal if $V_{\pi^*}(s) \geq V_\pi(s)$ for all s and for any other policy π, meaning that a higher Q value is preferred. As shown in Fig. 9(a), LCERL with the late-start strategy maintains consistently higher Q values, indicating more accurate value estimation and better anticipation of long-term rewards. In contrast, the LCERL without the late-start strategy shows low and flat Q value estimation, suggesting that poor early experiences misled the critic. This supports our theoretical claim that early low-quality experiences cause Q value underestimation, which hinders effective updates and slows down exploration.

Consequently, Fig. 9(b) shows that LCERL with the late-start strategy exhibits a higher loss in the early stages than its counterpart, which indicates that the algorithm is not misled by poor experiences. RL agent clearly recognizes that it is still far from the global optimal solution. Meanwhile, the loss value is stable, suggesting that LCERL with the late-start strategy is not trapped in a local optimum and continues to optimize the RL agent stably. In contrast, LCERL without the late-start strategy initially exhibits a lower loss, suggesting that it is misled by poor experiences and falls into a poor local optimum in the very beginning. After 600,000 training steps, it starts to explore more high-reward regions, but it is quite late.

Overall, Fig. 9 matches our theoretical analysis in Section III.E perfectly. Abundance of early low-quality experiences lead to the underestimation or incorrect estimation of the Q value, causing small or wrong ϵ_t in (7), influencing the parameter ω and θ update in (8) and (9), finally affecting the algorithm's training efficiency.

2) Double Opposite Proximal Mutation: To validate our assumption that the double opposite proximal mutation enhances the exploration ability of the algorithm, we compared the states that have been frequently visited by LCERL with and without the double opposite proximal mutation operator in Walker2d-v4, as shown in Fig. 10. Since there are 17 observation dimensions in Walker2d-v4, that are hard to display, we randomly selected two dimensions from the state vector and plotted kernel density estimation (KDE) graphs for these two dimensions to visualize the distribution of visited states. These plots act as "heat maps," showing areas that were frequently visited.

From the figure, we can see that LCERL with double opposite proximal mutation exhibits a larger area, which means it has explored more states. Importantly, LCERL does not sacrifice its exploitation ability in the process. The most frequently visited states of both algorithms, i.e., the orange areas, are similar around (0.0, -0.75), meaning they both find

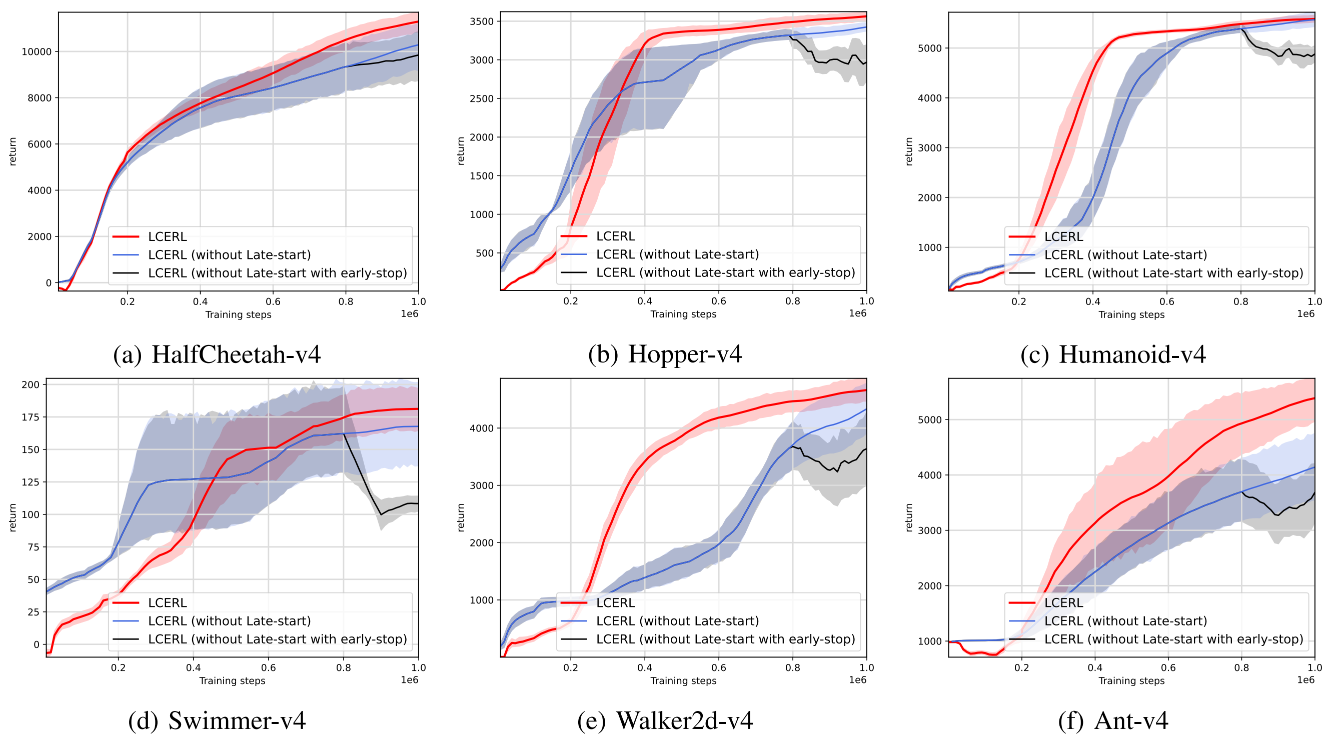


Fig. 6. Ablation experiment comparing the performance with and without the late start strategy. The red curve represents LCERL with the late start strategy, while the blue curve represents LCERL without the late start strategy. The black curve represents LCERL without the late start strategy with early stop.

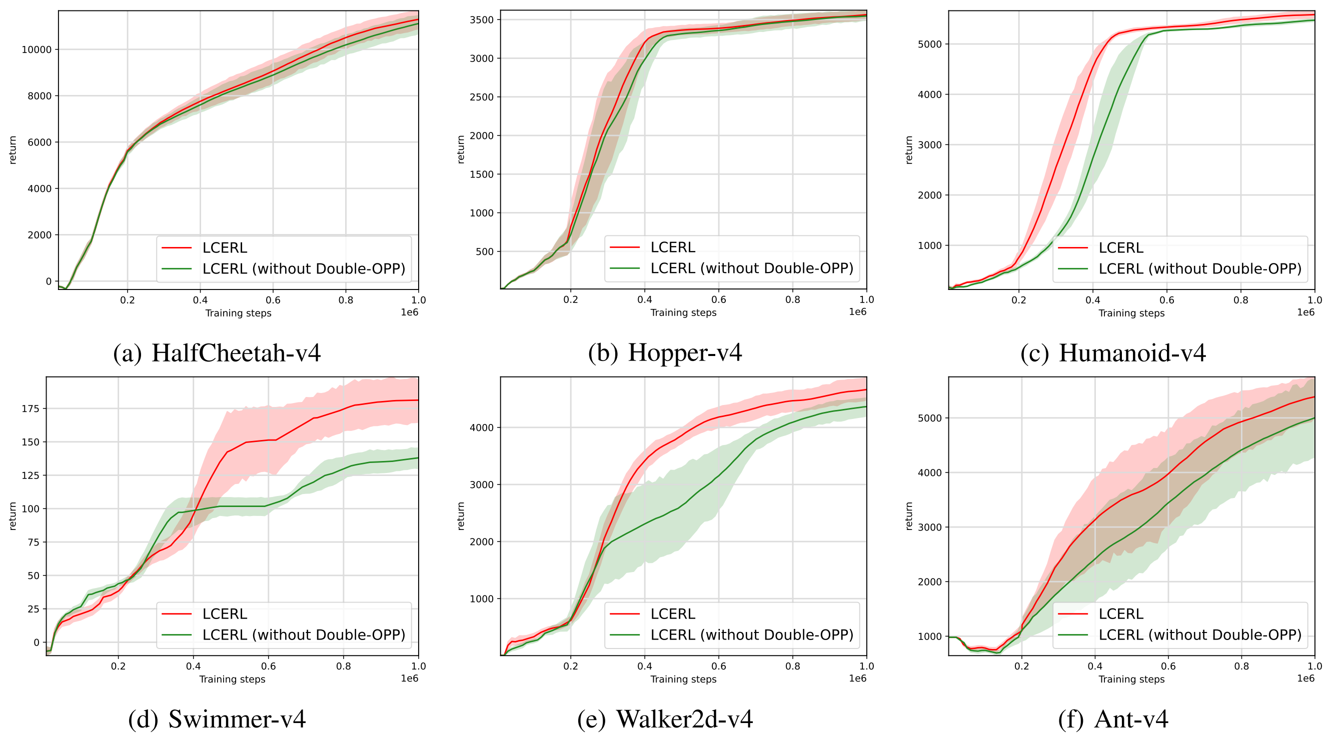


Fig. 7. Ablation experiment comparing the performance with and without the double opposite proximal mutation. The red curve represents LCERL with the double opposite proximal mutation, while the green curve represents LCERL without the double opposite proximal mutation methods.

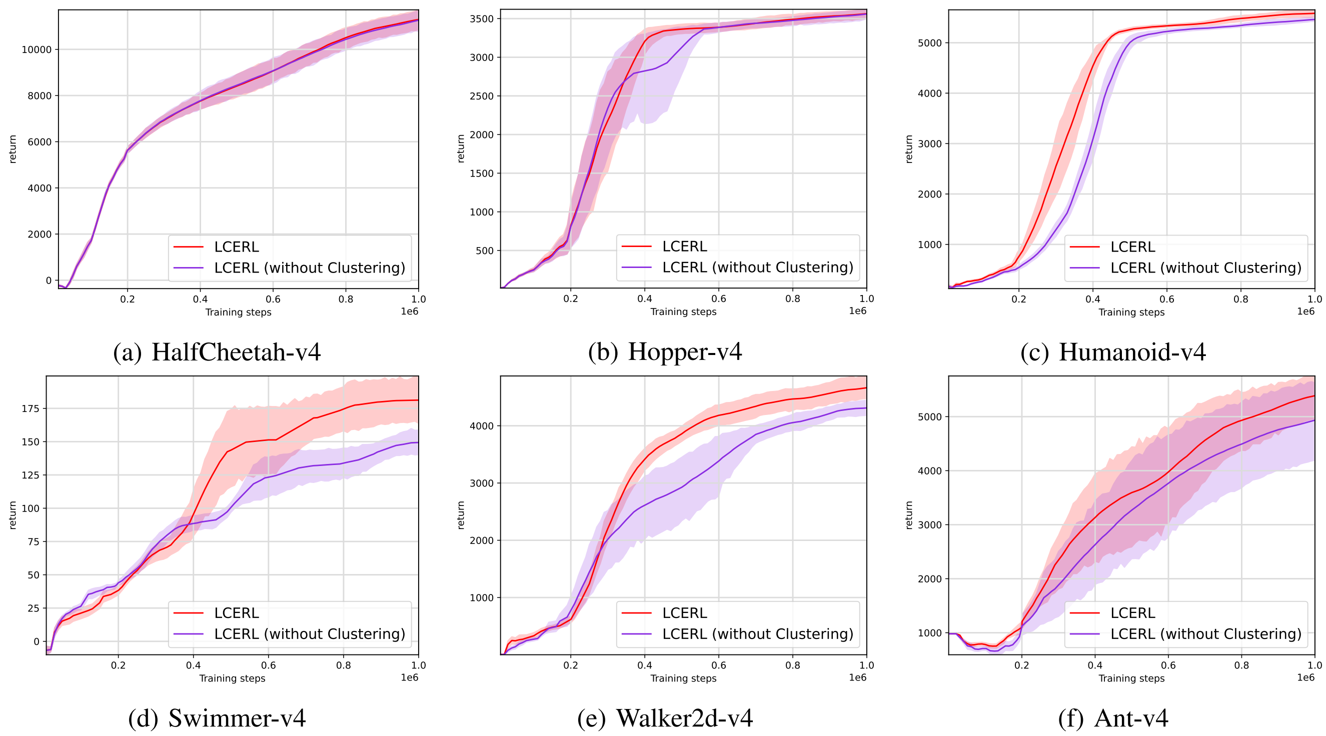


Fig. 8. Ablation experiment comparing the performance with and without the clustering. The red curve represents LCERL with the clustering, while the purple curve represents LCERL without the clustering.

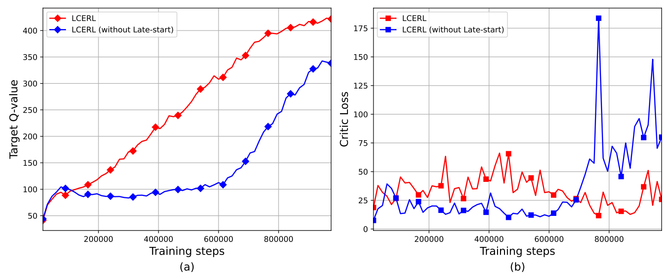


Fig. 9. The target Q value (a) and critic loss (b) of LCERL and LCERL without Late-start strategy.

these states with good values. The difference is that the orange area of LCERL with double opposite proximal mutation is larger, indicating that double opposite proximal mutation exploits a slightly larger neighborhood than proximal mutation. Meanwhile, Fig. 10(a) also shows that LCERL with double opposite proximal mutation also takes some effort to explore another promising state area around (0.0, 0.75). Overall, Fig. 10 showed that our algorithm effectively balanced exploration and exploitation capabilities, ensuring robust performance in diverse scenarios.

3) Clustering Selection with Archive: According to the ablation experimental results shown in Fig. 8, the effect of clustering selection on HalfCheetah-v4 and Hopper-v4 is not very obvious, but it is more obvious in the other four environments. We attribute this discrepancy to the inherent characteristics of each environment. Environments like Ant-v4 and Humanoid-v4 feature more complex locomotion and higher-dimensional state or action spaces, leading to more diverse policy behaviors. Walker2d-v4, although it has the same state and action dimensions as HalfCheetah-v4, is more complex since there are multiple ways to walk faster. As to Swimmer-v4, although it has simpler dynamics and lower action dimensionality than the others, according to research in [71], it often presents deceptive gradients, which can trap policies in poor local optima. In contrast, HalfCheetah-v4 and Hopper-v4 have a more unique and deterministic policy, where LCERL cannot find many good policies with different behaviors. In summary, the clustering selection with archive proves to be more effective in environments that either demand behavioral diversity or are prone to suboptimal convergence, while its effectiveness is limited in scenarios where the search space is unimodal, i.e., the optimal policy is quite singular.

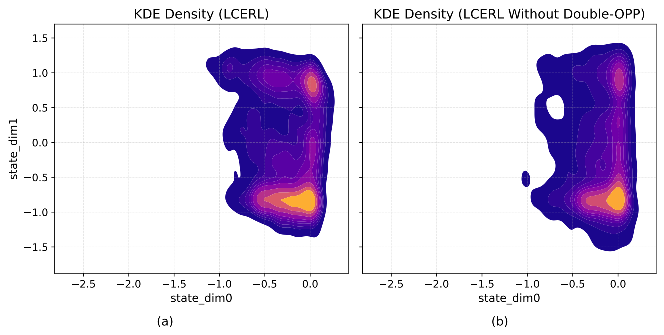


Fig. 10. The Kernel Density Estimation of (a) LCERL and (b) LCERL without double opposite proximal mutation.

TABLE V
COMPARISON OF TRAINING TIME (MINUTES) ACROSS ALGORITHMS

Classification	Method	HalfCheetah-v4	Hopper-v4	Humanoid-v4	Swimmer-v4	Walker2d-v4	Ant-4
RL	DDPG	183	290	880	257	254	306
	TD3	267	311	990	324	674	427
ERL	ERL	319	326	364	321	327	325
	CERL	4318	1065	844	1147	1865	4587
	CEM-RL	851	804	875	842	803	780
	PDERL	308	363	584	455	477	367
	ERL-Re^2	1289	1251	1703	1370	1396	1307
	ERL_TD	1210	1197	1744	1189	1442	1556
	TERL	351	382	499	361	366	387
	LCERL	389	398	644	431	502	445

E. Training Time

In this paper, All experiments were conducted on a system with Intel(R) Xeon(R) Silver 4310 CPU @ 2.10GHz with 12 cores, and NVIDIA RTX 4090D GPU. Training time may vary depending on implementation and hardware. Due to the double opposite proximal mutation having a certain probability of undergoing secondary mutation, it incurs additional time consumption. Therefore we compared LCERL with several representative ERL algorithms as well as high-performing RL algorithms, including ERL, CERL, CERL-RL, PDERL, ERL-Re^2, DDPG, and TD3. The results are presented in Table V. We can see that LCERL requires more time than RL algorithms such as DDPG and TD3, as well as the original ERL algorithm. This is a common characteristic of all ERL frameworks, as they require maintaining a population of RL actors, whereas RL algorithms only need to keep an RL agent. However, compared to recently proposed ERL algorithms, such as CERL, CEM-RL, and ERL-Re^2, LCERL demonstrates shorter training time. Meanwhile, due to the late-start strategy, the EA population is introduced at a later stage, which helps save computational time for our algorithm LCERL. This indicates that the proposed strategy does not incur additional time overhead compared to current ERL frameworks, thereby demonstrating the feasibility of our algorithm.

F. Experiments on MEMG scheduling problem

LCERL is compared with TD3 [24] and DDPG [48], which are two RL algorithms that were widely used for this problem. The dataset is derived from real-world scenarios, where thermal energy data is sourced from [72], and photovoltaic (PV) and electrical energy data are sourced from [73]. We collected one year of data and divided it into a training set consisting of 231 days and a test set consisting of 125 days. The training results are shown in Fig. 11, where it can be observed that LCERL outperforms TD3 and DDPG. Moreover, LCERL exhibits smaller standard deviations, indicating better stability. Since this study focuses on energy management in real-world environments, it is essential to evaluate the actual energy consumption and carbon emissions to verify the effectiveness of LCERL. We applied the trained actor to the test set, and the cumulative consumption on the test set in the real-world environment is shown in Fig. 12. The average consumption and total carbon emissions are summarized in Table VI. In the real-world MEMG environment, lower energy consumption is preferable. As shown in Fig. 12 and Table VI, LCERL achieves the lowest energy consumption and carbon emissions. Overall, LCERL not only performs well in the MuJoCo simulated environment but also demonstrates strong practicability on real-world applications.

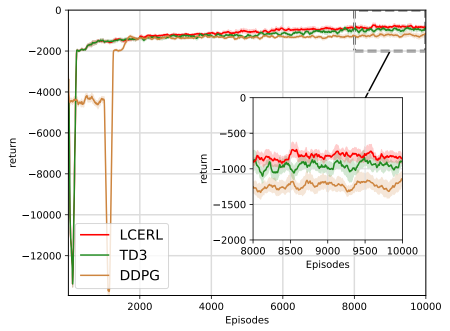


Fig. 11. The training curves on the MEMG environment.

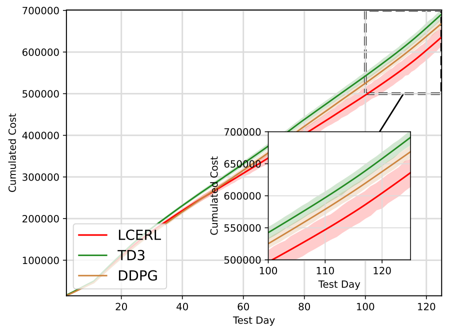


Fig. 12. The experiment results on the MEMG environment in the test stage.

TABLE VI
THE RESULT ON MEMG IN THE TEST STAGE

Algorithms	Cost(thous.$)	Carbon(t)
DDPG	557.45	8457.83
TD3	577.15	8144.64
LCERL	**531.93**	**7909.78**

V. CONCLUSION

The goal of this paper is to mitigate the negative influence of EA and enhance RL training efficiency in ERL, this goal has been successfully achieved by proposing the LCERL algorithm. It focused on generating diverse and high-quality experiences to enhance the synergy between EA and RL. Three key components, i.e., the late-start strategy, the double opposite proximal mutation operator, and the clustering selection method with archive, ensured the generation of high-quality actors and experiences. The experimental study has clearly demonstrated the advantages of LCERL against the state-of-the-art RL and ERL algorithms in convergence speed and stability. Meanwhile, a real-world energy scheduling application has shown its remarkable practicability.

Although our algorithm performs well, there are still some limitations. Firstly, the late-start strategy is currently controlled by a fixed hyperparameter δ. Future work will focus on exploring adaptive mechanisms based on the RL agent's learning progress. Secondly, while the double opposite proximal mutation improves actor performance, it may introduce additional computational overhead due to the secondary mutation step. Although experimental results show that this overhead is manageable, light surrogate models may be required in resource-constrained environments. Meanwhile, we will also parallelize the mutation evaluation and make full use of GPU acceleration in the future.

Finally, besides the MEMG problem that we have applied our algorithm to, it is promising to extend our algorithm to more real-world applications such as robotic control, smart grid optimization, and battery management. These fields face challenges such as uncertainty and noise in robotic control, high-dimensionality, delayed rewards, and sparse rewards in smart grid optimization, and the need for real-time decision-making in battery management. Leveraging its strong exploratory-exploitation capacity and robustness to sparse and delayed rewards, LERL is able to provide effective solutions in complex environments.

REFERENCES

[1] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, *An Introduction to Deep Reinforcement Learning*. Now Publishers, Inc., 2018.

[2] J. Jia and W. Wang, "Review of reinforcement learning research," in *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2020, pp. 186–191.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] 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.

[6] Z. Zhang, Z. Wu, H. Zhang, and J. Wang, "Meta-learning-based deep reinforcement learning for multiobjective optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7978–7991, 2022.

[7] W. Yi, R. Qu, L. Jiao, and B. Niu, "Automated design of metaheuristics using reinforcement learning within a novel general search framework," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 4, pp. 1072–1084, 2023.

[8] I.-B. Park and J. Park, "Scalable scheduling of semiconductor packaging facilities using deep reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 53, no. 6, pp. 3518–3531, 2023.

[9] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Niching genetic programming to learn actions for deep reinforcement learning in dynamic flexible scheduling," *IEEE Transactions on Evolutionary Computation*, 2024.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[11] M. Cai, Q. Wang, Z. Qi, D. Jin, X. Wu, T. Xu, and L. Zhang, "Deep reinforcement learning framework-based flow rate rejection control of soft magnetic miniature robots," *IEEE Transactions on Cybernetics*, vol. 53, no. 12, pp. 7699–7711, 2023.

[12] S. Kamio and H. Iba, "Adaptation technique for integrating genetic programming and reinforcement learning for real robots," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 318–333, 2005.

[13] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 11, pp. 15 051–15 071, Nov 2024.

[14] 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*, vol. 35, no. 9, pp. 11 939–11 957, 2024.

[15] W. Liu, R. Wang, T. Zhang, K. Li, W. Li, H. Ishibuchi, and X. Liao, "Hybridization of evolutionary algorithm and deep reinforcement learning for multiobjective orienteering optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 5, pp. 1260–1274, 2023.

[16] X. Zhou, A. K. Qin, M. Gong, and K. C. Tan, "A survey on evolutionary construction of deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 894–912, 2021.

[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," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[19] C. Bodnar, B. Day, and P. Lió, "Proximal distilled evolutionary reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3283–3290.

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

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

[22] Y. Tang, "Guiding evolutionary strategies with off-policy actor-critic," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 1317–1325.

[23] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, pp. 19–67, 2005.

[24] 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.

[25] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville, "The primacy bias in deep reinforcement learning," in *International conference on machine learning*. PMLR, 2022, pp. 16 828–16 847.

[26] 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.

[27] P. Li, J. Hao, H. Tang, X. Fu, Y. Zhen, and K. Tang, "Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms," *IEEE Transactions on Evolutionary Computation*, 2024.

[28] A. Leite, M. Candadai, and E. J. Izquierdo, "Reinforcement learning beyond the bellman equation: Exploring critic objectives using evolution," in *Artificial Life Conference Proceedings 32*. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ..., 2020, pp. 441–449.

[29] E. Marchesini and C. Amato, "Improving deep policy gradients with value function search," *arXiv preprint arXiv:2302.10145*, 2023.

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

[31] S. Elfwing, E. Uchibe, and K. Doya, "Online meta-learning by parallel algorithm competition," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 426–433.

[32] A. Sehgal, N. Ward, H. M. La, and S. Louis, "Deep reinforcement learning for robotic manipulation tasks using a genetic algorithm-based function optimizer," *Encyclopedia with semantic computing and robotic intelligence*, 2023.

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

[34] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: a new approach for hard-exploration problems," *arXiv preprint arXiv:1901.10995*, 2019.

[35] F.-Y. Liu, Z.-N. Li, and C. Qian, "Self-guided evolution strategies with historical estimated gradients." in *IJCAI*, 2020, pp. 1474–1480.

[36] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[37] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*. PMLR, 2019, pp. 2555–2565.

[38] H. Bharadhwaj, K. Xie, and F. Shkurti, "Model-predictive control via cross-entropy and gradient-based optimization," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 277–286.

[39] T. Mundhenk, M. Landajuela, R. Glatt, C. Santiago, D. Faissol, and B. Petersen, "Symbolic regression via neural-guided genetic programming population seeding. arxiv (2021)," *arXiv preprint arXiv:2111.00053*, 2021.

[40] M. I. Radaideh and K. Shirvan, "Rule-based reinforcement learning methodology to inform evolutionary algorithms for constrained optimization of engineering applications," *Knowledge-Based Systems*, vol. 217, p. 106836, 2021.

[41] Y. Song, L. Wei, Q. Yang, J. Wu, L. Xing, and Y. Chen, "Rl-ga: A reinforcement learning-based genetic algorithm for electromagnetic detection satellite scheduling problem," *Swarm and Evolutionary Computation*, vol. 77, p. 101236, 2023.

[42] W. Yi, R. Qu, L. Jiao, and B. Niu, "Automated design of metaheuristics using reinforcement learning within a novel general search framework," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 4, pp. 1072–1084, 2022.

[43] G. Cideron, T. Pierrot, N. Perrin, K. Beguir, and O. Sigaud, "Qd-rl: Efficient mixing of quality and diversity in reinforcement learning. corr abs/2006.08505 (2020)," *arXiv preprint arXiv:2006.08505*, 2020.

[44] M. Faldor, F. Chalumeau, M. Flageat, and A. Cully, "Map-elites with descriptor-conditioned gradients and archive distillation into a single policy," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 138–146.

[45] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–72, 1992.

[46] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer, 2004, vol. 133.

[47] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

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

[49] 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.

[50] 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.

[51] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. ieee, 1995, pp. 1942–1948.

[52] Y. Ma, T. Liu, B. Wei, Y. Liu, K. Xu, and W. Li, "Evolutionary action selection for gradient-based policy learning," in *International Conference on Neural Information Processing*. Springer, 2022, pp. 579–590.

[53] Q. Zhu, X. Wu, Q. Lin, and W.-N. Chen, "Two-stage evolutionary reinforcement learning for enhancing exploration and exploitation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, pp. 20 892–20 900, Mar. 2024.

[54] 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*. PMLR, 2019, pp. 3341–3350.

[55] F. Espositi and A. Bonarini, "Gradient bias to solve the generalization limit of genetic algorithms through hybridization with reinforcement learning," in *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part I 6*. Springer, 2020, pp. 273–284.

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

[57] K. Suri, "Off-policy evolutionary reinforcement learning with maximum mutations," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1237–1245.

[58] H. Zheng, P. Wei, J. Jiang, G. Long, Q. Lu, and C. Zhang, "Cooperative heterogeneous deep reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 17 455–17 465.

[59] X. Wu, Q. Zhu, Q. Lin, W. Chen, and J. Li, "Adaptive evolutionary reinforcement learning algorithm with early termination strategy," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024, pp. 1947–1955.

[60] Y. Wang, T. Zhang, Y. Chang, X. Wang, B. Liang, and B. Yuan, "A surrogate-assisted controller for expensive evolutionary reinforcement learning," *Information Sciences*, vol. 616, pp. 539–557, 2022.

[61] H. Jianye, P. Li, H. Tang, Y. Zheng, X. Fu, and Z. Meng, "Erl-re 2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation," in *The Eleventh International Conference on Learning Representations*, 2022.

[62] C. Hu, J. Liu, and X. Yao, "Evolutionary reinforcement learning via cooperative coevolution," in *ECAI 2024*. IOS Press, 2024, pp. 3300–3307.

[63] Q. Chen, B. Xue, and M. Zhang, "Preserving population diversity based on transformed semantics in genetic programming for symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 433–447, 2020.

[64] E. Burke, S. Gustafson, and G. Kendall, "A survey and analysis of diversity measures in genetic programming," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, 2002, pp. 716–723.

[65] 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.

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

[67] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

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

[69] 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.

[70] Y. Wang, K. Xue, and C. Qian, "Evolutionary diversity optimization with clustering-based selection for reinforcement learning," in *International Conference on Learning Representations*, 2021.

[71] N. Kim, H. Baek, and H. Shin, "Pgps: Coupling policy gradient with population-based search," in *International Conference on Learning Representations*, 2020.

[72] 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.

[73] 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.

List of Figures

List of Tables