Neural Processing Letters (2024) 56:69
https://doi.org/10.1007/s11063-024-11548-6

Adaptive Evolutionary Reinforcement Learning with Policy Direction

Caibo Dong[1] · **Dazi Li**[1]

Accepted: 28 January 2024 / Published online: 23 February 2024

Abstract
Evolutionary Reinforcement Learning (ERL) has garnered widespread attention in recent years due to its inherent robustness and parallelism. However, the integration of Evolutionary Algorithms (EAs) and Reinforcement Learning (RL) remains relatively rudimentary and lacks dynamism, which can impact the convergence performance of ERL algorithms. In this study, a dynamic adaptive module is introduced to balance the Evolution Strategies (ES) and RL training within ERL. By incorporating elite strategies, this module leverages advantageous individuals to elevate the overall population's performance. Additionally, RL strategy updates often lack guidance from the population. To address this, we incorporate the strategies of the best individuals from the population, providing valuable policy direction. This is achieved through the formulation of a loss function that employs either L1 or L2 regularization to facilitate RL training. The proposed framework is referred to as Adaptive Evolutionary Reinforcement Learning (AERL). The effectiveness of our framework is evaluated by adopting Soft Actor-Critic (SAC) as the RL algorithm and comparing it with other algorithms in the *MuJoCo* environment. The results underscore the outstanding convergence performance of our proposed Adaptive Evolutionary Soft Actor-Critic (AESAC) algorithm. Furthermore, ablation experiments are conducted to emphasize the necessity of these two improvements. It is worth noting that the enhancements in AESAC are realized at the population level, enabling broader exploration and effectively reducing the risk of falling into local optima.

Keywords Adaptive evolutionary reinforcement learning · Adaptive evolutionary soft actor-critic · Soft actor-critic · Policy direction

1 Introduction

Reinforcement Learning (RL) [1] stands as a prominent subfield within machine learning, focused on training agents to acquire optimal behavior strategies through interactions with the environment. In RL, agents receive state information from the environment, take actions

✉ Dazi Li
lidz@mail.buct.edu.cn

[1] College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China

based on the current state, and receive rewards or punishments as feedback for their actions. RL has been found extensive applications in diverse domains such as games [2], robot control [3], and optimization [4–6]. In recent years, with the rapid development of Deep Reinforcement Learning (DRL), many high-performance algorithms have been proposed, such as Deep Deterministic Policy Gradient (DDPG) [7], Proximal Policy Optimization (PPO) [8], and Soft Actor-Critic (SAC) [9]. However, these algorithms still face challenges related to hyperparameter sensitivity and limited convergence performance. The SAC algorithm, recognized for its exceptional sample efficiency, operates as a maximum entropy RL algorithm that adeptly balances exploration and exploitation objectives. Nevertheless, certain stochastic conditions can still impede robustness and yield suboptimal training outcomes in RL problems [10]. However, Evolutionary Algorithms (EAs) [11], due to their inherent capacity to explore global optimal solutions, hold promise in mitigating this limitation within the realm of RL.

Evolutionary Algorithms (EAs), derived from imitating biological evolution mechanisms, have found broad application in domains such as optimization and scheduling [12]. EAs are distinguished by their robustness and stability, which stem from the diversity of individuals within their populations. In contrast to RL [13], EAs provide a gradient-free optimization solution and demonstrate favorable convergence properties, particularly when executed in highly parallel computing environments. However, this advantage comes at the cost of significant computational resources. EAs tend to converge towards global optimal solutions. Nonetheless, the assessment of policy quality necessitates a full iteration, resulting in lower sample efficiency and a limited exploration approach [14]. Some RL algorithms, owing to their high sample efficiency and the capacity to promptly provide gradient information through single-step updates, can mitigate the shortcomings of EAs.

The combination of Evolutionary Algorithms (EAs) and Reinforcement Learning (RL) has been extensively studied in recent years, as it allows for addressing the respective deficiencies of each approach. EAs can explore and learn within the parameter space, while RL operates in the action space. ERL can be categorized into two main types: non-feedback ERL and feedback ERL. In non-feedback ERL [15], the learning processes of EAs and RL are almost separate, limiting their effectiveness. Feedback ERL [16, 17] primarily integrates the gradient information of RL into the EAs. As one of the pioneering feedback ERL algorithms, the Evolutionary Reinforcement Learning (a specialized algorithm) [18] combines the exploration capabilities of EAs with the sample efficiency of RL. Several algorithms within the ERL framework have been proposed, either by replacing or augmenting components of EAs and RL segments, such as the Evolution-based Soft Actor-Critic (ESAC) algorithm [19] which exhibits similar performance to the SAC. Nevertheless, there is still considerable room for enhancing the performance of ERL. Moreover, there are relatively limited researches on the combination method of EAs and RL [20], and it is difficult to balance learning and exploration between the two. Exploring the relationship between these two approaches and leveraging their combined strengths to further enhance performance is an area that warrants more attention and investigation. Additionally, the timing of when to initiate RL training significantly impacts the overall training outcomes, yet existing researches are typically based on fixed-step methods. Current approaches of RL individuals injecting gradient information into EAs lack distinction, while emphasizing the utilization of useful gradient information is essential. The adaptive module proposed in this article can highlight the dominant individuals of the population in the ERL and retain more gradient information of the dominant RL individuals within the population, especially when combined with the elite strategy [21]. Moreover, the direction provided by EAs to RL is still insufficient. The proposed policy direction method employs the optimal individuals within the population for

RL parameter guidance, expediting the RL learning process without any adverse effects. Considering the sensitivity of EAs to mutation rates [22] and the necessity of continuous exploration during the early stage of the learning process, an exponential decay mutation rate is designed to facilitate robust exploration capabilities for both EAs and RL within the parameter space and action spaces during the initial stages.

In the context of our research, Adaptive Evolutionary Reinforcement Learning (AERL) with policy direction is proposed. The core innovation lies in considering the proportion of RL individuals (individuals correspond to policy parameters) within the population and determining when to introduce gradient information from RL training into the population during the evolutionary reinforcement learning processes. This dynamic and adaptive process fosters a harmonious synergy between the two learning modalities, enabling individuals within the population to adapt to a high-performance state. Additionally, as RL may not incorporate the learning achievements of the population in its own learning, we address this limitation by guiding RL individuals with the optimal individuals within the population. These optimal individuals actively contribute to the RL training process through the utilization of a loss function, without introducing any adverse effects on the RL training. Finally, recognizing the sensitivity of mutation rates, stronger exploratory behavior is preferred during the early stages of evolutionary training. To achieve this, mutation rates are updated by using an exponential decay approach. We selected SAC as RL algorithm to introduce the AESAC algorithm for experimental verification and highlighted its good performance.

2 Background and Related Work

2.1 Evolution Strategies

Evolutionary Strategy (ES) [23] is a parameter optimization method inspired by biological evolution. It has emerged as a prominent black-box optimization method. Specifically, the Natural ES [24] is used, whose genes are composed of real numbers obtained via sampling from the Gaussian distribution to generate new individuals. Most EAs tend to discard the majority of solutions and retain only the best one [12]. However, those poorer solutions often contain valuable information beneficial for computing better parameter estimates in subsequent generations. In Natural Evolution Strategies (NES), information from all individuals within the population, regardless of their quality, is utilized to update parameters. Then, the parameters are updated through the fitness of the individuals in the environment. The next generation population consists of individuals with new parameters. Similar to conventional EAs, natural ES involves essential elements such as crossover, mutation, and selection.

Previous researches have explored the possibility of utilizing natural ES as a substitute for RL algorithms, demonstrating competitive performance. However, the sample efficiency of natural ES is hindered by the requirement of evaluating individual fitness after each iteration. To expedite the training process, multiple workers can be employed in parallel to speed up the training process. Although parallelization is highly efficient, it does entail substantial computational resource consumption. In our study, we extensively investigate the potential benefits of employing parallel natural ES to accelerate the learning process. And the combination of maximum entropy reinforcement learning yields a noteworthy improvement in sample efficiency [25].

Focusing on the parameter optimization, ϕ are defined as the parameters of the actor network of the RL agent, which is used to make decisions. $F(\cdot)$ is the fitness function composed

of the return from the environment within a single round. The population distribution P_ς is instantiated as the Gaussian distribution $N(\mu, \sigma^2)$ with the mean μ and the standard deviation σ. The average fitness of each parameter can be written as $E_{\phi\sim P_\varsigma} F(\phi)$. Generally, we directly set $E_{\phi\sim P_\varsigma} F(\phi) = E_{\varepsilon\sim N(0,1)} F(\phi+\lambda\varepsilon)$. λ is the mutation rate. In practical operations, each worker is defined to sample noise from the normal distribution $N(0, 1)$ with different random numbers. Generally, the individuals obtained by sampling the population are hoped to have higher fitness, and the parameter θ are updated using gradient descent, resulting in the achievement of the expression:

$$\nabla_\phi E_{\phi\sim P_\varsigma} F(\phi) = \nabla_\phi E_{\varepsilon\sim N(0,1)} F(\phi+\lambda\varepsilon) = 1/\sigma E_{\varepsilon\sim N(0,1)}\{F(\phi+\lambda\varepsilon)\varepsilon\} \tag{1}$$

Parallel Natural Evolution Strategies (ES for short) can efficiently utilize distributed computing resources, thereby accelerating the optimization process. This enables them to tackle large-scale problems and discover solutions through parallel computation within a constrained timeframe [26]. For parallel evolutionary learning, the process can be divided into two main steps. The first step involves multiple workers interacting with the environment to evaluate the effectiveness of perturbed parameters, ultimately acquiring scalar fitness values. In the second step, the parameters are updated based on the obtained fitness values and their corresponding perturbations. Algorithm 1 below demonstrates a simple implementation of this process.

Algorithm 1 Parallel Natural Evolution Strategies

Initialize mutation rate λ ,learning rate β_{ES} ,initial policy parameters θ
for generation =1, ∞ do
  for each worker i =1, …, n do
    sample the random seed and sample $\varepsilon_i \sim N(0,1)$
    get the fitness F_i by evaluating the $(\theta+\lambda\varepsilon_i)$
  end for
  update $\theta \leftarrow \theta + \beta_{ES} / n\sigma E_{\varepsilon\sim N(0,1)}\{F_i\varepsilon_i\}$
end for

From the Algorithm 1, it can also be observed that when obtaining fitness values F_i, there is a need to evaluate policy parameters $(\theta+\lambda\varepsilon_i)$. Nonetheless, even with the parallel approach, achieving this still necessitates interacting with the environment for an entire episode and assessing the cumulative reward. This approach inherently leads to lower sample efficiency.

2.2 Maximum Entropy Reinforcement Learning

RL is a methodology for acquiring rewards through the interaction of an agent with its environment and maximizing the rewards to obtain an optimal strategy. Typically, RL problems in continuous space are modeled as Markov decision processes (MDP) [27], consisting of a tuple (S, A, P, r), where the state space S and action space A are continuous, P represents the state transition probability, and r is a bounded reward function given by the environment feedback. The agent follows a policy and uses ρ_π to represent the trajectory distribution. The Soft Actor Critic (SAC) algorithm, within the maximum entropy framework, leverages off-policy capabilities for the reutilization of past experiences. Additionally, this framework effectively balances exploration and exploitation. Furthermore, the algorithm employs

single-step updates, which are advantageous for promptly introducing gradient information into the population. Unlike traditional RL, the objective function of the maximum entropy reinforcement learning algorithm SAC is as follows:

$$J(\pi) = \sum_{t=0}^{T} E_{(s_t, a_t)\sim\rho_\pi}[r(s_t, a_t) + \eta\mathcal{H}(\pi(\cdot|s_t))] \tag{2}$$

where $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy term with the coefficient η. SAC evolved from soft policy iteration and has strong convergence performance, with powerful exploration ability in the early stage to avoid local optima. The algorithm initializes the state value function V_ϖ, Q value function Q_δ, policy output $\pi_\phi(a_t|s_t)$, and their corresponding network parameters ϖ, δ, ϕ. The loss function of the state value network is:

$$J_V(\varpi) = E_{s_t\sim D}\left[\frac{1}{2}\left(V_\varpi(s_t) - E_{a_t\sim\pi_\phi}\left[Q_\delta(s_t, a_t) - \log\pi_\phi(a_t|s_t)\right]\right)^2\right] \tag{3}$$

where D represents an experience pool. The loss function of the Q network is the Bellman residual, given by:

$$J_Q(\delta) = E_{(s_t, a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\delta(s_t, a_t) - \hat{Q}(s_t, a_t)\right)^2\right] \tag{4}$$

with $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1}\sim p}[V_\varpi(s_{t+1})]$ where γ is the discount rate. By minimizing the expected KL-divergence [28], the policy network parameters are updated as:

$$J_\pi(\phi) = E_{s_t\sim\mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}(\pi_\phi(\cdot|s_t)\Big\|\frac{\exp(Q_\delta(s_t, \cdot))}{Z_\delta(s_t)}\right] \tag{5}$$

Through the acquisition of the loss function and the execution of gradient backpropagation to update the parameters, the algorithm eventually attains convergence. In our framework, SAC serves as the RL algorithm for AERL.

Policy parameters ϕ are updated through the computation of the loss function as defined in equal (5). It's important to note that if the initial policy selection is inappropriate, it can result in the algorithm exploring in the vicinity of local optima. Furthermore, in certain complex environments replete with numerous local optima, relying solely on RL algorithms may not suffice in ensuring robust performance. Therefore, considering the integration of Evolutionary Strategies (ES) to enhance algorithmic performance becomes a compelling proposition.

3 Method

3.1 Adaptive Evolutionary Reinforcement Learning (AERL)

Incorporating the policy parameters updated by RL agents into the population of ES is a crucial step in feedback ERL. However, the collaborative dynamics between ES and RL training, as well as the optimization of their respective performances, have not been comprehensively explored. This study aims to investigate the coupling relationship between ES and RL, encompassing aspects such as the proportion of RL individuals within the population and the opportune moment for introducing the gradient information of RL. This dynamic adjustment process unfolds throughout the training of feedback ERL, thereby attaining adaptive states in response to the evolving training conditions of ES and RL. Significantly, when combined with elite strategy, the advantageous individual parameters of both ES and RL can

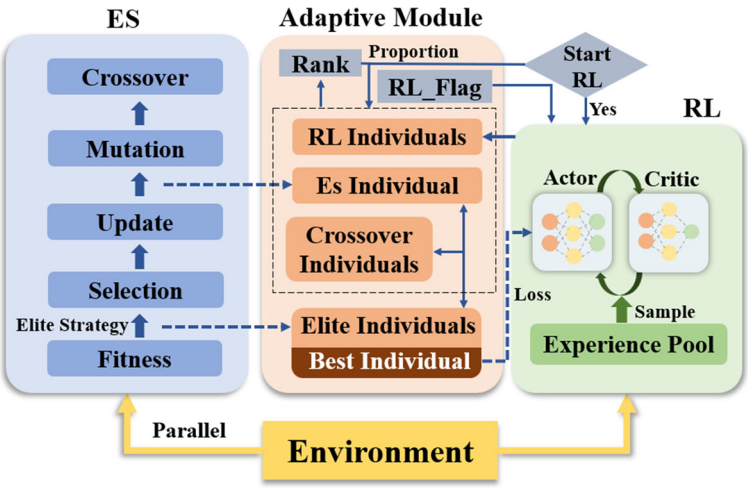

Fig. 1 The flow chart of our proposed framework AERL

be preserved. The flow chart of our proposed framework AERL is depicted in Fig. 1, and the detailed execution of the algorithm is expounded upon below. The AERL framework, as illustrated in the diagram, comprises three primary components: ES, RL, and Integration. Our primary innovations are concentrated within the Integration component, encompassing the introduction of an adaptive module designed to harness individual strengths effectively and policy direction for utilizing the top-performing individual within the population to guide RL training.

The initialization of population P is defined by composing the initial parameters of ES individuals, RL individuals, and crossover individuals. Crossover individuals are generated through the crossing of ES individuals and elite individuals. The proportion of crossover individuals within the population is fixed at 40%. Initially, all individuals' parameters within the population are perturbed and evaluated through interactive simulations with the environment. Multiple workers are deployed to facilitate this process, ultimately yielding corresponding fitness values. The parameters of elite individuals and the best individual ϕ_{best} are obtained from the evaluation results. The best individual ϕ_{best} will be operated in updating RL individual.

In the adaptive module, the fitness of the individuals within the population is first ranked, such as the rank of RL individuals is r_1, $r_2 ... r_n$, $n = N_{RL}$, and when the population size is N_{ALL}, the number of RL policy individuals N_{RL} in the next generation population is

$$N_{\mathrm{RL}} = clip\left(\chi\left(N_{ALL} - \frac{\sum_{i=1}^{N_{RL}} r_i}{N_{RL}}\right), \ell_{\min} * N_{ALL}, \ell_{\max} * N_{ALL}\right) \tag{6}$$

where χ is the individual learning coefficient that is utilized to enhance or diminish the proportion of RL individuals within a population. $\ell_{\min}$ is the lower bound coefficient of truncation and $\ell_{\max}$ is the is the upper bound coefficient. Establishing upper and lower limits ensures that the RL individuals is not too weak within the population or too strong, resulting in a too small proportion of ES individuals. It helps avert scenarios where ES training and RL training are temporarily neglected due to poor performance.

The advantage of this approach lies in its ability to dynamically adjust the proportion of RL individuals within the population based on the performance of RL training. When RL training yields superior results, this adaptive mechanism increases the presence of RL individuals, thereby expediting the overall training process of ERL. However, if the performance of RL individuals falls short of the optimum, ES individuals and crossover individuals take on more significant roles, ensuring that the training process remains robust. In essence, this represents an internal adaptive control mechanism operating within the population, optimizing its composition in response to the prevailing training conditions.

Based on the calculated N_{RL}, external adaptive control can be performed. The purpose of RL training is to introduce the learned parameters, that is, gradient information, into the population. If the ranking of the RL individuals is high when $N_{\mathrm{RL}} > \upsilon_b * N_{ALL}$ with υ_b the RL threshold coefficient, it means that RL plays a leadership role. When combined with crossover, the individuals will have an advantage transfer with the elite individuals, mainly based on the RL individuals. At this time, intensifying RL training can accelerate individuals optimization within the population. Conversely, if RL individuals are ranked poorly, the emphasis can temporarily shift towards ES training, with elite individuals predominantly derived from ES individuals. This approach fully capitalizes on the complementary strengths of ES and RL, with one operating within the parameter space and the other within the action space. The introduction of the *RL_Flag* is designed to maintain a consistent number of RL training steps throughout each ERL training process. The purpose of introducing condition $RL\_Flag = \upsilon_{\max} * N_{ALL}$,with $\upsilon_{\max}$ as the RL starting coefficient, to initiate RL training is to avoid the continuous training of ES when the RL training results are suboptimal. This continuous ES training could impede the mutual complementarity of the two approaches. This external adaptive control is controlled by N_{RL} and *RL_Flag*.

3.2 Policy Direction

In ERL, RL individuals often lack guidance from ES during training. Given that evaluation occurs in a population-based form, ES can derive an optimal policy from the current population, which holds instructive significance for RL training. In the AERL, the method of policy direction is adopted, which entails learning directly by imitating the optimal strategy in the parameter space. It is clear to calculate a distance by comparing the population's optimal policy with the current RL policy (policy is equals to individual) to update the RL policy. The expression representing the distance is denoted as $D(\phi_{\mathrm{RL}}, \phi_{best})$, where ϕ_{RL} is the current RL policy and ϕ_{best} is the population's optimal policy. Since the guideline does not be supposed to keep changing during the update, only the best individual at that time $\overline{\phi}_{best}$ is imported.

L1 [29] or L2 norms [30] are commonly employed metrics to quantify distances. We can utilize either L1 or L2 norms to measure the distance between two sets of neural network parameters and guide policy by minimizing this distance. If the L1 norm is selected, its expression is as follows:

$$D(\phi_{\mathrm{RL}}, \phi_{best}) = \kappa_1 ||\phi_{\mathrm{RL}} - \phi_{best}||_1 = \kappa_1 \sum_{i=1}^{n} |\phi_{\mathrm{RL},i} - \phi_{best,i}| \tag{7}$$

where κ_1 is the L1 coefficient.

If L2 norm, the expression is:

$$D(\phi_{\mathrm{RL}}, \phi_{best}) = \kappa_2 ||\phi_{\mathrm{RL}} - \phi_{best}||_2 = \kappa_2 \sqrt{\sum_{i=1}^{n} (\phi_{\mathrm{RL},i} - \phi_{best,i})^2} \tag{8}$$

where κ_2 is the L2 coefficient.

Combined with the distance metric, the loss function for updating the SAC policy is as follows:

$$L(\phi_{\mathrm{RL}}) \cdot E_{s_t\sim\mathcal{D},\varepsilon_t\sim\mathcal{N}}[\log \pi_{\phi_{\mathrm{RL}}}(f_{\phi_{\mathrm{RL}}}(\varepsilon_t; s_t)|s_t) - Q_\delta(s_t,\ f_{\phi_{\mathrm{RL}}}(\varepsilon_t; s_t))] + D(\phi_{\mathrm{RL}},\ \phi_{best}) \tag{9}$$

where ε_t is the noise vector which is sampled from a normal distribution N. $f_{\phi_{\mathrm{RL}}}$ is the implicit policy network.

The expression of the derivative of the total loss to ϕ_{RL} is below:

$$\begin{aligned}&\frac{\partial L(\phi_{\mathrm{RL}})}{\partial \phi_{\mathrm{RL}}} \cdot \nabla_{\phi_{\mathrm{RL}}} \log \pi_{\phi_{\mathrm{RL}}}(a_t|s_t) \\ &+ \left(\nabla_{\mathbf{a}_t} \log \pi_{\phi_{\mathrm{RL}}}(a_t|s_t) - \nabla_{\mathbf{a}_t} Q(s_t,\ a_t)\right) \nabla_{\phi_{\mathrm{RL}}} f_{\phi_{\mathrm{RL}}}(\varepsilon_t; s_t) + \frac{\partial D(\phi_{\mathrm{RL}},\ \phi_{best})}{\partial \phi_{\mathrm{RL}}}\end{aligned} \tag{10}$$

where a_t is sampled from the $f_{\phi_{\mathrm{RL}}}$.

For the case of L1,

$$\frac{\partial D(\phi_{\mathrm{RL}},\ \phi_{best})}{\partial \phi_{\mathrm{RL},i}} = \frac{\kappa_1 \partial \sum_{i=1}^{n} |\phi_{\mathrm{RL},i} - \phi_{best,i}|}{\partial \phi_{\mathrm{RL},i}} = \kappa_1 \mathrm{sign}\left(\phi_{\mathrm{RL},i} - \phi_{best,i}\right) \tag{11}$$

For L2,

$$\frac{\partial D(\phi_{\mathrm{RL}},\ \phi_{best})}{\partial \phi_{\mathrm{RL},i}} = \frac{\kappa_2 \partial \sqrt{\sum_{i=1}^{n} \left(\phi_{\mathrm{RL},i} - \phi_{best,i}\right)^2}}{\partial \phi_{\mathrm{RL},i}} = \frac{\kappa_2(\phi_{\mathrm{RL},i} - \phi_{best,i})}{\sqrt{\sum_{i=1}^{n}\left(\phi_{\mathrm{RL},i} - \phi_{best,i}\right)^2}} \tag{12}$$

Due to the fact that a single training episode of RL may involve up to a thousand steps, consistently using the optimal parameter individual to guide the reinforcement learning process might diminish the advantage of RL during the later stages of the episode. Therefore, the guidance of the optimal individual is only employed during the first one hundred steps of the RL process.

3.3 Adaptive Evolutionary Soft Actor-Critic (AESAC)

In order to instantaneously investigate the integration and coupling relationship between ES and RL in AERL, a novel algorithm called Adaptive Evolutionary Soft Actor-Critic (AESAC) is proposed. It includes an adaptive module that can dynamically adjust the proportion of RL (SAC) individuals within the population and the opportune moment of SAC training based on the training situation of ES and SAC. Additionally, we incorporate policy direction methods to accelerate the learning process of SAC individuals towards the optimal individual within the population, without introducing any negative impact as SAC individuals are also included in the population.

In the standard maximum entropy evolutionary reinforcement learning algorithm ESAC, the number of SAC individuals within the population remains constant, and the initiation of SAC training depends solely on a fixed number of steps, disregarding any population-based information. However, ES primarily explores the parameter space, while SAC algorithms delve into the policy space. It is pivotal to adaptively increase the proportion of SAC individuals within the population and to allow SAC training to persist when its performance is relatively favorable, thereby leveraging on the strengths of SAC. The elitist strategy preserves the best individuals from the previous generation and enables their advantageous parameters

to be retained by crossing over with ES individuals. Our adaptive module can effectively harness the influential role of these advantageous individuals in conjunction with the elitist strategy. Moreover, guidance from the best individual within the population introduces a bias during SAC gradient updates. By adopting L1 or L2 regularization, the discrepancy between the SAC individual and the best individual within the population can be reduced, thus facilitating convergence during training.

To speed up the convergence of the algorithm, an exponential decay method is adopted to update the mutation rate. During the early stage of the algorithm, a larger mutation rate is required to enhance the diversity of the population and facilitate exploration in both the parameter space of ES and the policy space of SAC. As the learning process advances, it becomes crucial to decrease the mutation rate in order to accelerate convergence. The specific expression for the mutation rate is determined as follows:

$$\lambda = \lambda_{final} + \left(\lambda_{start} - \lambda_{final}\right) * e^{-\frac{n_{step}}{n_{decay}}} \tag{13}$$

where λ_{final} is the mutation rate in the end of the training, λ_{start} is the mutation rate in the beginning. n_{step} is the immediate number of training steps, n_{decay} is the number of all training steps.

The specific algorithm is shown in Algorithm 2.

Algorithm 2 Adaptive Evolutionary Soft Actor-Critic (AESAC)

Initialize $\lambda, \beta_{ES}, \ell_{\max}, \ell_{\min}, \chi, \upsilon_b, \upsilon_{\max}, \kappa_1, \kappa_2, \eta, \gamma$

initialize parameters vectors $\phi_{ES}, \phi_{RL}, \delta, \varpi, \bar{\varpi}$

initialize the number of all individuals N_{ALL}, SAC individuals N_{RL} and the flag of RL *RL_Flag*

initialize the population with *n* actors

for generation =1, ∞ do

  for each worker i =1, …, *n* do

    sample the random seed and sample $\varepsilon_i \sim N(0,1)$

    get the fitness F_i by evaluating the $(\phi_{ES} + \lambda\varepsilon_i)$

  end for

  rank the F_i and get the average rank of SAC individuals

  calculate the N_{RL} of next generation according to Eq. (6).

  if $N_{RL} > \upsilon_b * N_{ALL}$ or $RL\_Flag = \upsilon_{\max} * N_{ALL}$:

    Obtain the current best individual $\bar{\phi}_{best}$ within the population

    update the ϕ_{RL} with the policy direction according to Eq. (10).

    update the $\delta, \varpi, \bar{\varpi}$ according to SAC algorithm

    initial *RL_Flag*

  else:

    $RL\_Flag += 1$

  update $\phi_{ES} \leftarrow \phi_{ES} + \beta / n\lambda E_{\varepsilon \sim N(0,1)}\{F_i \varepsilon_i\}$

  update λ according to exponential decline function Eq. (13).

  Composition of the next generation from N_{RL} SAC Individuals, ϕ_{ES} individuals, crossover individuals

end for

3.4 Guideline to Develop Similar Algorithms

In the AERL framework I propose comprises ES, RL algorithm, adaptive module, and policy direction. ES is often recognized for its capacity to provide global optimal solutions and exhibit stronger robustness, although it tends to be less time-efficient. Conversely, RL is a highly sample-efficient algorithm with exploration capability, yet it may become trapped in local optima. Within our adaptive module, based on the learning performance of RL and ES, we can fully harness the strengths of these individuals. RL individuals are integrated as part of the ES population. Concurrently, policy direction refers to the utilization of the best individuals from the ES to guide the RL training. This framework allows for the development of similar algorithms, where ES can be replaced with other evolutionary algorithms. Regarding the reinforcement learning algorithm, we have chosen the currently well-converging algorithm SAC, but other reinforcement learning algorithms can also be employed.

4 Experiments

4.1 Comparative Evaluation

To validate the effectiveness of our proposed AESAC algorithm, continuous control tasks from MuJoCo [31] based on the open AI are conducted, which are widely-recognized benchmark for evaluating continuous RL algorithms. MuJoCo (Multi-Joint dynamics with Contact) is a leading physics simulation engine used for modeling complex dynamics. This simulation environment provides a dynamic and realistic setting for training and evaluating intelligent agents in RL. MuJoCo's efficient physics simulation engine facilitates the development and testing of RL algorithms, making it a valuable tool for academic research in the field of machine learning and artificial intelligence. The HalfCheetah, Hopper, Walker2d, and Swimmer used in the experiments are different simulation environments provided by MuJoCo. These MuJoCo tasks encompass diverse challenges in motion control. In the HalfCheetah task, the goal is to efficiently propel a half-cheetah model forward to maximize rewards. The Hopper task involves orchestrating a one-legged robot to execute forward jumps while maintaining balance. For the Walker2d task, the objective is to achieve swift and stable forward walking with a bipedal robot. Lastly, in the Swimmer task, a three-link swimmer model is controlled using two joints to achieve rapid forward swimming. Each of these tasks poses unique demands on control and coordination, making them valuable benchmarks for assessing the performance of RL algorithms.

In our algorithm, the network structures are shown in Fig. 2. The network architectures for RL individuals and ES individuals share two fully connected layers. However, there is a divergence when it comes to policy outputs: ES utilizes a Mean Linear Layer 3 to derive actions, whereas RL requires both Mean Layer 3 and Std Layer 3 for action sampling. The structure of the RL Critic network segment consists of three fully connected layers. The activation function employed in the neural network is Rectified Linear Unit (ReLU), and the dimension of hidden layers is 256.

Specific parameter settings in the experiments are shown in Table 1. These parameters are manually selected within the applicable range. It's noteworthy that SAC algorithm effectively mitigates hyperparameter sensitivity, simplifying the process of parameter selection.

Experiments are performed with five different random seeds to ensure reliable results. The hardware used for the experiments includes NVIDIA RTX3090 GPU and i9-12900 K

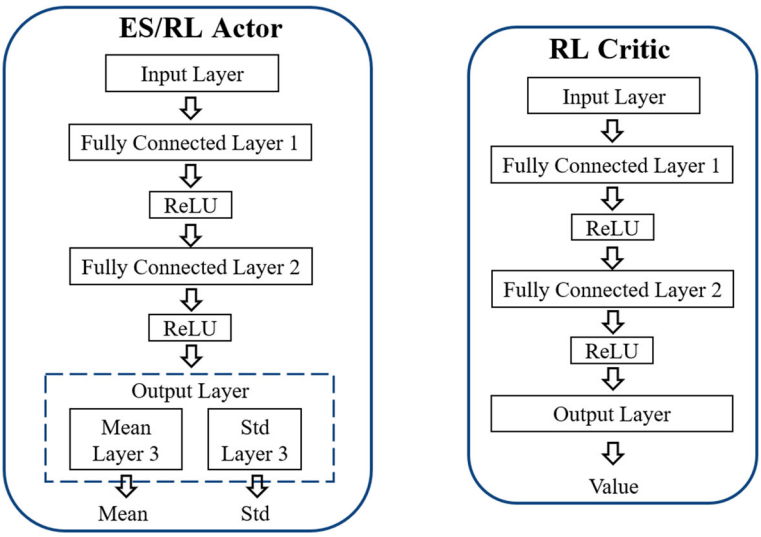

Fig. 2 The ES/RL individuals network structure and Critic network structure in RL

Table 1 The parameter settings

Description	Symbols	Values
Initial mutation rate	λ	0.06
Number of all individuals	N_{ALL}	10
Initial number of RL individuals	N_{RL}	4
ES learning rate	β_{ES}	0.005
RL learning rate	β_{RL}	0.0003
Upper bound coefficient of truncation	$\ell_{\max}$	0.5
Lower bound coefficient of truncation	$\ell_{\min}$	0.2
Individual learning coefficient	χ	1
RL threshold coefficient	υ_b	0.3
RL start coefficient	$\upsilon_{\max}$	0.6
L1 coefficient	κ_1	0.01
L2 coefficient	κ_2	0.01
Discount rate	γ	0.99

CPU. AESAC is compared with ESAC, SAC, DDPG, and PPO algorithms. The settings for the common SAC algorithm part in AESAC, ESAC, and SAC are the same. The policy networks for the SAC part use the same Gaussian policy, and both the policy and value networks consisted of fully connected layers. The results of the experiments are depicted in Fig. 3, where the shaded areas of different colors represent the range between the best and worst training results of each algorithm, and the solid lines represent the average results from five training runs. It can be seen that in the HalfCheetah, Hopper, Walker2d, and Swimmer environments, AESAC outperforms the other algorithms in terms of average training results, with superior best and worst training results. Except for the Hopper environment,

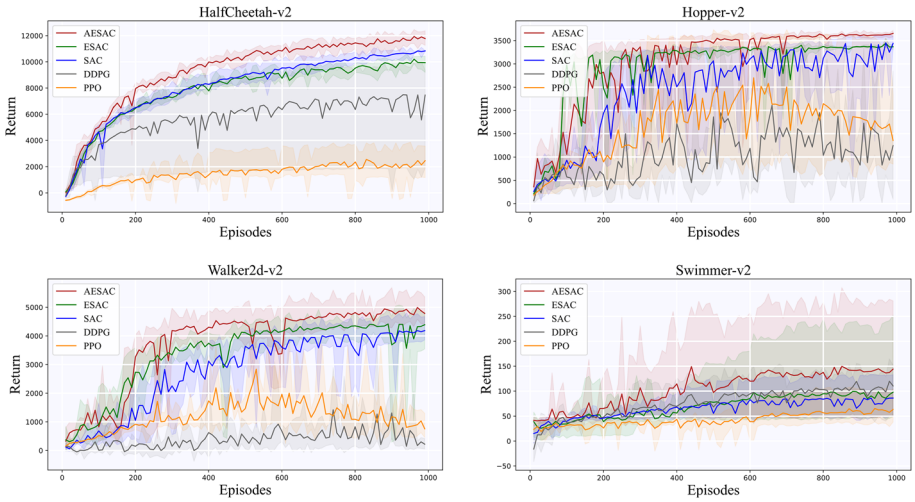

Fig. 3 The training curves of five algorithms (AESAC, ESAC, SAC, DDPG, PPO) on four Mujoco tasks

where ESAC's convergence speed is better than that of AESAC, AESAC outperforms the other four algorithms in terms of both convergence speed and final convergence value in the remaining three environments. When comparing AESAC, ESAC, and SAC, it was observed that ESAC and SAC displayed similar performance in the four environments, with ESAC not surpassing SAC in terms of convergence value, even falling behind SAC in the HalfCheetah environment. Additionally, ESAC don't perform better than SAC in terms of convergence speed either. Therefore, our proposed AESAC algorithm demonstrates superior performance, which compensates ESAC's shortcoming of not surpassing SAC. The minimum, maximum, and average values of the five algorithms when they converged in the four environments are summarized in Table 2. The highlighted data represents the highest values among the

Table 2 The convergence results of five algorithms (AESAC, ESAC, SAC, DDPG, PPO) on four Mujoco tasks

Task		AESAC	ESAC	SAC	DDPG	PPO
HalfCheetah-v2	Min	11,070.60	8916.28	10,627.31	1889.94	1247.59
	Max	12,468.11	10,458.94	11,026.27	9586.76	3691.85
	Mean	11,806.57	9770.95	10,837.55	7437.77	2196.55
Hopper-v2	Min	3568.31	3321.77	2530.38	753.32	851.42
	Max	3662.67	3389.21	3609.90	2766.85	3378.22
	Mean	3629.53	3360.87	3250.03	1425.73	1739.07
Walker2d-v2	Min	4467.97	3550.59	3826.44	227.51	511.47
	Max	5451.12	4830.14	4696.97	439.35	981.50
	Mean	4882.81	4408.56	4156.62	329.91	727.04
Swimmer-v2	Min	47.67	49.02	46.57	46.16	37.56
	Max	275.47	248.11	130.31	132.02	103.29
	Mean	141.91	96.00	75.46	102.55	60.05

convergence results of the five algorithms. It can be observed that, except for the case where AESAC has a lower minimum value than ESAC in the Swimmer environment, ESAC outperforms the other algorithms in all other cases. This demonstrates the superior performance of AESAC. Analyzing the average convergence values of the algorithms, it is evident that AESAC exhibits an improvement of approximately 10% compared to ESAC in the Hopper and Walker2d environments, a 20% improvement in the HalfCheetah environment, and a significant 50% improvement in the Swimmer environment. The superiority of AESAC over ESAC can be attributed primarily to the improvements made in the adaptive module and policy direction. DDPG and PPO algorithms display noticeably inferior performance compared to SAC, which leverages the advantages of maximum entropy reinforcement learning [32, 33], enabling it to avoid local optima and exhibit good convergence properties. In the next section, we will discuss the ablation experiments conducted to further investigate the contributions of the adaptive module and policy direction components in AESAC.

4.2 Ablation Study

The previous section presents experimental results that showcased the exceptional performance of AESAC in continuous control tasks compared to the other four algorithms, particularly outperforming ESAC. In comparison to ESAC, the key improvements in AESAC involve the incorporation of the adaptive module and policy direction. To further investigate the effectiveness of these components and determine if their combination yields superior results compared to each component alone, ablation experiments are conducted in the HalfCheetah environment. The experiments are repeated with five random seeds, maintaining the same experimental settings as described in the previous section. The specific results of this ablation experiment are illustrated in Fig. 4.

As shown in Fig. 4, the ESAC with the adaptive module achieves higher rewards compared to the ESAC, confirming the significance of the adaptive module in AESAC and its positive impact on improving the integration between ES and SAC. However, due to the lack of

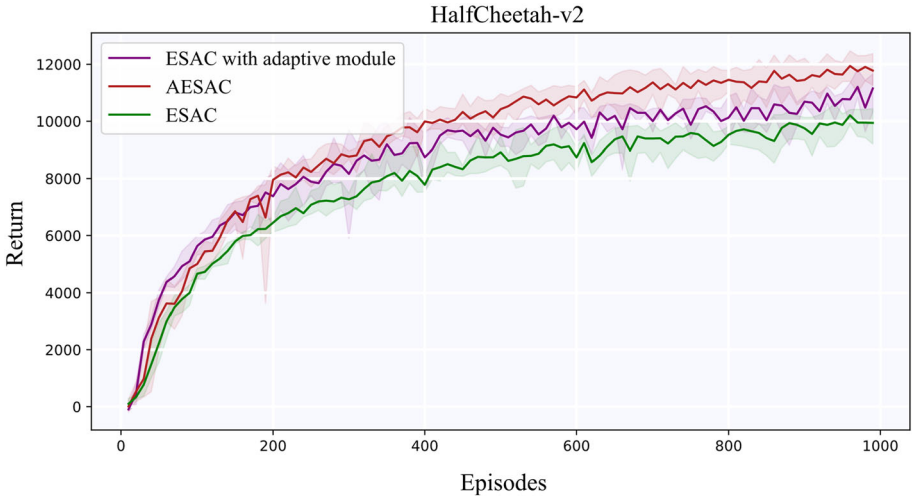

Fig. 4 The training curves of the ESAC with the adaptive module, AESAC, and ESAC

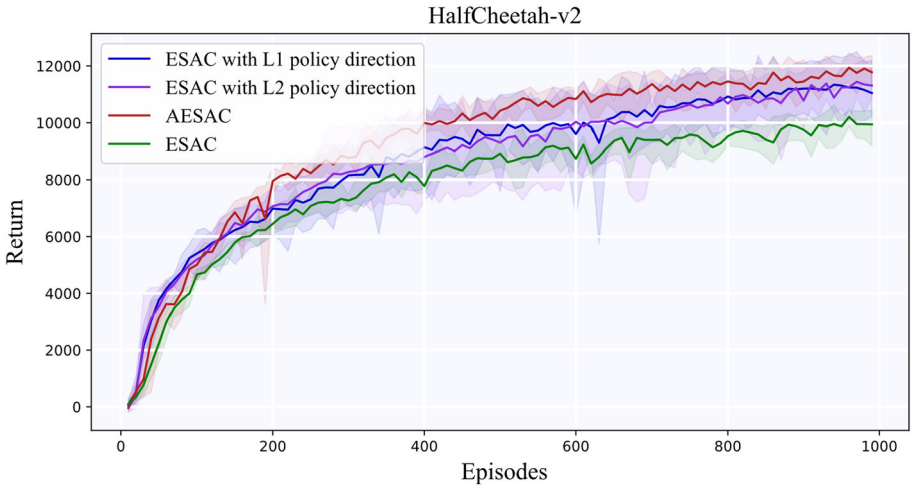

Fig. 5 The training curves of the ESAC with policy direction, AESAC, and ESAC

policy direction, the performance of ESAC with the adaptive module still falls short of the AESAC. This highlights the crucial role of policy direction in updating the SAC policy. In our experiments, L1 and L2 regularization are employed to improve the original ESAC. The results of these experiments are shown in Fig. 5.

Analysis of Fig. 5 reveals that both L1 and L2 regularization techniques applied to ESAC result in performance improvements. Specifically, the AESAC adopts L2 regularization. A comparison of the training results between AESAC and ESAC enhanced by L2 policy direction indicates that the combination of the adaptive module on the ESAC with L2 policy direction is beneficial. Therefore, these findings confirm the effectiveness of adding the adaptive module or policy direction into ESAC, as well as the effectiveness of incorporating one improvement after another.

4.3 Analysis of Individuals Within Populations

In the learning process of ERL, the population consists of ES individuals, RL individuals, as well as crossover individuals. In the AESAC, we have made improvements not only focusing on the best individuals within the population but also considering the performance within the whole population. Worst individuals and the average performance of the population are considered. This comprehensive evaluation enables us to better assess the effectiveness of algorithmic improvements and their impact on the overall population dynamics.

Figure 6 displays the plotted best, worst, and average returns of individuals within the population during the training process of the AESAC and ESAC algorithms. Each line represents the best, worst, or average performance, and different colors represent AESAC and ESAC algorithms. From the experimental results, it can be observed that AESAC consistently outperforms the ESAC in terms of the best individual, worst individual, and average policy within the population. Therefore, it is clear that the improvement of AESAC over ESAC is not limited to the performance of a few individuals within the population but extends to the population as a whole. This is mainly attributed to the effective utilization of advantages offered by the adaptive module and the iterative learning in both parameter space and policy space.

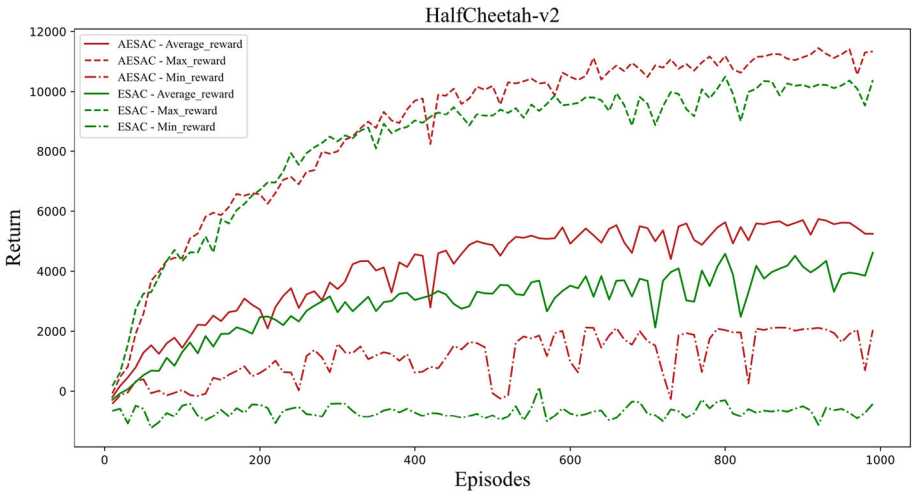

Fig. 6 The training curves of the best individual, worst individual, and average policy within the population between AESAC and ESAC

Furthermore, the SAC training is guided by the parameters of the best individuals, leading to the observed improvements in SAC. Importantly, it should be noted that the SAC individuals themselves are integral members of the population, further emphasizing the significance of their contributions in driving the overall improvement in AESAC.

4.4 The Analysis of Parameters Diversity

In AESAC, ES concentrates on exploring the parameter space, while SAC algorithms explore the policy space. The combination of the adaptive module and policy direction fully maximizes the advantages of both ES and SAC. It ensures a diverse range of parameters, particularly during the early stages of training. To analyze and compare the parameter diversity of the policy individuals, we employ dimensionality reduction techniques such as TSNE and visualization [34] to analyze the weights of the first fully connected layer of the ES policy and RL policy in AESAC and ESAC. First, compare the parameter diversity of SAC individuals between the AESAC and ESAC. The visualization results presented in Fig. 7 display a subset of training points during the early stages of training. From Fig. 7, it can be observed that the AESAC exhibits a wider exploration range in the parameters of SAC individuals. The results are closely related to the effective utilization of the adaptive module. Additionally, the policy direction incorporated in the SAC training process introduces more possibilities and expands the parameter training space of SAC.

Similarly, a comparison of the parameter diversity among the ES individuals in AESAC and ESAC is conducted. The visualization results, depicted in Fig. 8, show a subset of training points during the early stages of training. The parameter diversity in AESAC is found to be broader, indicating improved exploratory behavior in the ES individuals. This observation further emphasizes the connection between the superior performance of AESAC compared to ESAC and the enhanced exploratory nature of individuals. Moreover, the larger parameters space of SAC individuals and ES individuals in AESAC reduces the likelihood of getting trapped in local optima.

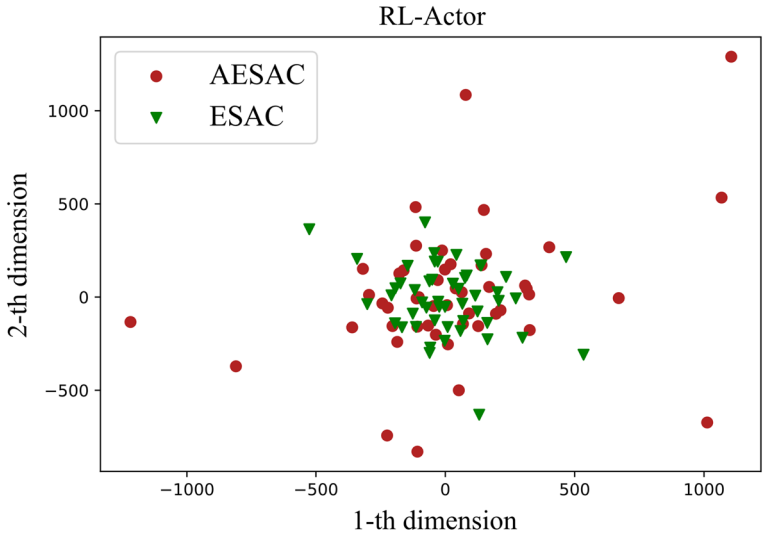

Fig. 7 The training distribution diagram of some parameters of reinforcement learning in the early stage of training after TSNE dimensionality reduction

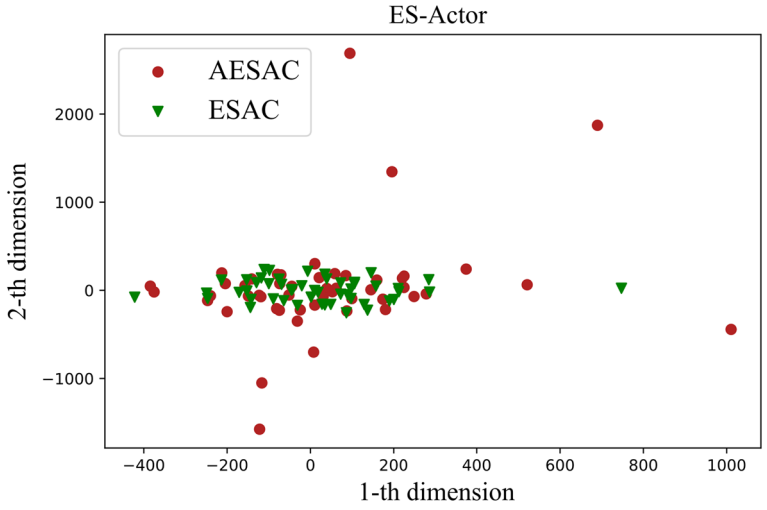

Fig. 8 The training distribution diagram of some parameters of the evolutionary strategy in the early stage of training after TSNE dimensionality reduction

4.5 The Analysis of Computing Complexity

In AERL, the definition of problem complexity is related to the number of individuals N_{ALL} within the population and the number of iterations N_I. Each individual within the population needs to undergo policy evaluation, crossover, mutation, and parameter updates. As previously analyzed, the total training steps in RL are essentially the same as the total number of

Table 3 The time complexity comparison of AESAC, ESAC, SAC, and ES

Algorithms	Execution time (h)	Proportion
ESAC	25.98	1
SAC	5.81	0.22
ES	19.95	0.77
AESAC	31.36	1.21

iterations in ES. Therefore, the computational complexity of this algorithm can be defined using Big O notation as $O(N_{ALL} * N_I)$.

In this part, we experimentally verify the time complexity of the proposed algorithm compared to other algorithms. By breaking down the time complexity of our algorithm AESAC into three parts, namely evolutionary strategy, SAC, adaptive module and policy direction, we provide a comparative analysis of AESAC against ESAC, SAC, and ES algorithms in terms of time complexity, as shown in Table 3, all after running 1000 episodes in the HalfCheetah environment.

The time complexity of ESAC is taken as 1, and the time complexities of the other three algorithms are marked for comparison. Notably, the RL algorithm SAC constitutes only a small fraction of the population. Hence, the time complexity of SAC is expected to be lower than ESAC, at approximately 0.22. Conversely, using the ES directly, as compared to ESAC, reduces the training portion of reinforcement learning, resulting in a time complexity of approximately 0.77 compared to ESAC. Moreover, it is noticeable that the algorithm execution time for ES approaches the sum of the execution time of SAC, which closely approximates the execution time of ESAC algorithm. Our algorithm builds upon the existing ESAC algorithm by introducing an adaptive module and policy direction. Due to the setting of *RL_Flag*, the reinforcement learning training steps in our algorithm are the same as in ESAC, and the evolutionary strategy training steps are also identical. The primary difference in time complexity arises during policy direction, where the norm of the network parameters of the best individual within the population needs to be computed. It is noteworthy that, compared to ESAC, our algorithm exhibits a 20% increase in time complexity, yet it yields a 10–20% performance improvement.

5 Conclusion

In this study, the AERL is introduced, which incorporates improvements in the form of an adaptive module and policy direction on the ERL. In our investigation of the combination of ES and RL, the proposed adaptive module adjusts the number of RL individuals within the population adaptively and determines when to initiate RL training by the *RL_Flag* parameter, which facilitates the advantages of ERL during the learning process. To incorporate the parameter information of the population's best individual into RL training, policy direction is introduced. It reduces the discrepancy between the RL individuals and the best individual within the population using L1 or L2 regularization, without introducing any detrimental effects. To verify the AERL framework, AESAC is proposed by incorporating the SAC algorithm into the AERL framework. Experimental results demonstrate that AESAC outperforms ESAC, SAC, and four other algorithms in terms of learning speed and convergence. In addition, ablation experiments are conducted to validate the effectiveness of each improvement.

Moreover, the training results of the best, worst, and average individuals within the population emphasize that the algorithmic improvements in AESAC operate at the population level and demonstrate stronger robustness. Furthermore, in the experiments on parameter diversity, AESAC demonstrates stronger parameter diversity compared to ESAC. This facilitates avoiding local optima and leads to performance improvement. In the time complexity experiment, AESAC exhibits higher time complexity, despite demonstrating better convergence performance. This is also the limitation of our algorithm. In the future, we can explore alternative approaches that offer higher time efficiency. This study has shed light on the combination method of ES and RL and demonstrated improvements in the performance of ERL. It provides promising research directions for future investigations, including more efficient combination methods, methods of utilizing RL gradient information to guide ES update, among others.

Acknowledgements This work was substantially supported by the National Natural Science Foundation of China under Grants 62273026.

Author Contributions Caibo Dong, writing & experiments Dazi Li, Configuration & revision.

Declarations

Competing interests The authors declare no competing interests.

References

1. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction, 2nd edn. The MIT Press, Cambridge, MA, USA
2. Lample G, Chaplot DS (2017) Playing FPS games with deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, pp 2140–2146
3. Nguyen H, La H (2019) Review of deep reinforcement learning for robot manipulation. In: 2019 Third IEEE international conference on robotic computing (IRC), pp 590–595
4. Ming Z, Zhang H, Li W, Luo Y (2023) Base on $ Q $-learning Pareto optimality for linear Itô stochastic systems with Markovian jumps. IEEE Trans Autom Sci Eng 1–11
5. Zhang W, Ji M, Yu H, Zhen C (2023) ReLP: reinforcement learning pruning method based on prior knowledge. Neural Process Lett 55(4):4661–4678
6. Yang Y, He J, Chen C, Wei J (2023) Balancing awareness fast charging control for lithium-ion battery pack using deep reinforcement learning. IEEE Trans Ind Electron 1–10
7. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971
8. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
9. Haarnoja T, Zhou A, Abbeel P, Levine S (2018) Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning (PMLR), pp 1861–1870
10. Xu T, Yang Z, Wang Z, Liang Y (2021) Doubly robust off-policy actor-critic: convergence and optimality. In: International conference on machine learning (PMLR), pp 11581–11591

11. Slowik A, Kwasnicka H (2020) Evolutionary algorithms and their applications to engineering problems. Neural Comput Appl 32:12363–12379
12. Sheng M, Chen S, Liu W, Mao J, Liu X (2022) A differential evolution with adaptive neighborhood mutation and local search for multi-modal optimization. Neurocomputing 489:309–322
13. Salimans T, Ho J, Chen X, Sidor S, Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864
14. Such FP, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J (2017) Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567
15. Colas C, Sigaud O, Oudeyer PY (2018) Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In: International conference on machine learning (PMLR), pp 1039–1048
16. Pourchot A, Sigaud O (2018) CEM-RL: Combining evolutionary and gradient-based methods for policy search. arXiv preprint arXiv:1810.01222
17. Lü S, Han S, Zhou W, Zhang J (2021) Recruitment-imitation mechanism for evolutionary reinforcement learning. Inf Sci 553:172–188
18. Khadka S, Tumer K (2018) Evolution-guided policy gradient in reinforcement learning. Adv Neural Inf Process Syst 31:1–13
19. Suri K, Shi XQ, Plataniotis KN, Lawryshyn YA (2020) Maximum mutation reinforcement learning for scalable control. arXiv preprint arXiv:2007.13690
20. Drugan MM (2019) Reinforcement learning versus evolutionary computation: a survey on hybrid algorithms. Swarm Evol Comput 44:228–246
21. Dulebenets MA (2020) Archived elitism in evolutionary computation: towards improving solution quality and population diversity. Int J Bio Inspir Comput 15(3):135–146
22. Lehre PK, Qin X (2022) Self-adaptation via multi-objectivisation: a theoretical study. In: Proceedings of the genetic and evolutionary computation conference, pp 1417–1425
23. Hussien AG, Heidari AA, Ye X, Liang G, Chen H, Pan Z (2023) Boosting whale optimization with evolution strategy and Gaussian random walks: an image segmentation method. Eng Comput 39(3):1935–1979
24. Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J, Schmidhuber J (2014) Natural evolution strategies. J Mach Learn Res 15(1):949–980
25. Wang J, Lei S, Liang L (2020) Preparation of porous activated carbon from semi-coke by high temperature activation with KOH for the high-efficiency adsorption of aqueous tetracycline. Appl Surf Sci 530:147187
26. Ben-Nun T, Hoefler T (2019) Demystifying parallel and distributed deep learning: an in-depth concurrency analysis. ACM Comput Surv CSUR 52(4):1–43
27. Li J, Ren T, Yan D, Su H, Zhu J (2022) Policy learning for robust Markov decision process with a mismatched generative model. In: Proceedings of the AAAI conference on artificial intelligence, pp 7417–7425
28. Joyce JM (2011) Kullback–Leibler divergence. International encyclopedia of statistical science
29. Li LT, Li DZ, Song TH, Xu X (2020) Actor-critic learning control with regularization and feature selection in policy gradient estimation. IEEE Trans Neural Netw Learn Syst 32(3):1217–1227
30. Li LT, Li DZ, Song TH, Xu X (2018) Actor-critic learning control based on $\ell_ 2 $-regularized temporal-difference prediction with gradient correction. IEEE Trans Neural Netw Learn Syst 29(12):5899–5909
31. Todorov E, Erez T, Tassa Y (2012) Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ International conference on intelligent robots and systems, pp 5026–5033
32. Shi W, Song S, Wu C (2019) Soft policy gradient method for maximum entropy deep reinforcement learning. arXiv preprint arXiv:1909.03198
33. Chen P, Pei J, Lu W, Li M (2022) A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. Neurocomputing 497:64–75
34. Cai TT, Ma R (2022) Theoretical foundations of t-sne for visualizing high-dimensional clustered data. J Mach Learn Res 23(1):13581–13634