A Surrogate-Assisted Controller for Expensive Evolutionary Reinforcement Learning

Yuxing Wang[a], Tiantian Zhang[a], Yongzhe Chang[a], Bin Liang[b], Xueqian Wang[a] and Bo Yuan[a,*]

[a]Shenzhen International Graduate School, Tsinghua University, Shenzhen, 518055, China
[b]Department of Automation, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Evolutionary algorithm
Evolutionary reinforcement learning
Surrogate model

ABSTRACT

The integration of Reinforcement Learning (RL) and Evolutionary Algorithms (EAs) aims at simultaneously exploiting the sample efficiency as well as the diversity and robustness of the two paradigms. Recently, hybrid learning frameworks based on this principle have achieved great success in various challenging robot control tasks. However, in these methods, policies from the genetic population are evaluated via interactions with the real environments, limiting their applicability in computationally expensive problems. In this work, we propose Surrogate-assisted Controller (SC), a novel and efficient module that can be integrated into existing frameworks to alleviate the computational burden of EAs by partially replacing the expensive policy evaluation. The key challenge in applying this module is to prevent the optimization process from being misled by the possible false minima introduced by the surrogate. To address this issue, we present two strategies for SC to control the workflow of hybrid frameworks. Experiments on six continuous control tasks from the OpenAI Gym platform show that SC can not only significantly reduce the cost of fitness evaluations, but also boost the performance of the original hybrid frameworks with collaborative learning and evolutionary processes.

1. Introduction

Reinforcement Learning (RL) has demonstrated promising achievements in various domains, ranging from Atari games [1], GO [2], to robot control tasks [3]. Among these successes, Deep Learning (DL) techniques [4] such as Deep Neural Networks (DNNs) have been widely used for decision-making [5, 6, 7]. The combination of RL with DL is generally called Deep Reinforcement Learning (DRL). However, recent studies show that DRL suffers from premature convergence to local optima in the training process, and the RL agents are highly sensitive to hyperparameter settings, implementation details and uncertainties of the environmental dynamics [8], preventing agents from learning stable policies.

In the meantime, black-box optimization techniques such as Evolutionary Algorithms (EAs) [9, 10] have shown competitive results compared to DRL algorithms [11]. Firstly, the population-based mechanism makes EAs explore the parameter space better than DRL. Secondly, since EAs only consider the total returns across the entire episode, they are indifferent to the issue of sparse reward and robust to environmental noise [11]. However, EAs suffer from low sample efficiency [12], due to their inherent black-box properties, and they often do not make full use of the feedback signals and historical data from the environment.

An emergent research direction is dedicated to exploiting the benefits of both solutions following the theory of evolution [13], where the learning of individuals can increase the evolutionary advantage of species, which can subsequently make the population learn faster [14]. Recently proposed Evolutionary Reinforcement Learning (ERL) [12], Proximal Distilled ERL (PDERL) [15] and other hybrid frameworks based on EA and off-policy DRL [16, 17, 18] have demonstrated encouraging progresses on well-known tasks with large continuous state and action spaces [19], outperforming pure RL and EA approaches.

However, there is a notable issue with these hybrid frameworks: the evaluation of the genetic population requires all individuals to be presented to the environment

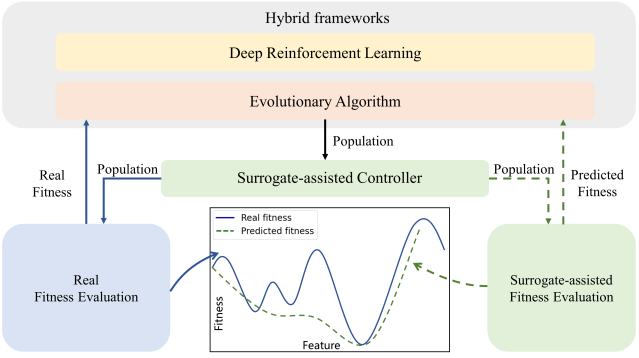


Figure 1: An overview of Surrogate-assisted Controller (SC). The SC switches between the real fitness function and the approximated fitness function constructed by the surrogate model to efficiently evaluate the genetic population in hybrid frameworks, while reducing the computational cost of evolutionary optimization and boosting the performance of the original hybrid frameworks. As shown at the bottom, in some cases, inaccurate surrogates may bring extra benefits: the predicted fitness function (dashed curve) has large deviations from the real one (solid curve) but nevertheless features the same global minimum and constructs a new fitness landscape that is more friendly to evolutionary search.

*Corresponding author at: The Center for Artificial Intelligence and Robotics, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China. E-mail address:boyuan@ieee.org.
ORCID(s):

during the training process. Although interactions can provide diverse experiences for DRL, they usually require a large amount of computational time and can be prohibitively expensive [20, 21]. For instance, in robotic scenarios, frequent evaluations may lead to massive resources consumption or even equipment damage. These unfavorable characteristics greatly limit the applicability of existing hybrid frameworks in complex simulated circumstances or non-trivial real-world scenarios.

A possible solution is to introduce the mechanism of surrogate. In typical Surrogate-assisted Evolutionary Optimization (SEO), surrogates, also known as meta-models, are trained to partially replace the expensive fitness functions [22]. Surrogate models such as kernel-based Kriging [23], polynomials [24] and neural networks have been successfully applied to domains including constrained optimization [25] and multi-objective optimization [26]. Nevertheless, in the standard RL context, due to the large uncertainties of genotype-phenotype-fitness mapping [27], conventional methods face great challenges of high computational cost and modeling complexity (Section 2.2) with only very limited studies conducted on simple discrete control tasks.

In this paper, we design a simple yet effective module, called Surrogate-assisted Controller (SC), that can be conveniently combined with existing hybrid RL frameworks to improve their practicability. As shown in Figure 1, SC employs an approximated fitness function together with the real fitness function to help evaluate the genetic population. It is computationally efficient and its surrogate model can be easily implemented in the existing hybrid RL frameworks, making full use of the diverse experiences to evaluate the fitness of individuals without environmental interactions. Note that the primary criterion for applying the surrogate model is the introduced prediction error [28]. SC mitigates this risk with two management strategies with elite protection mechanism (Section 4.2), which can strategically schedule the real and the surrogate-assisted evaluation as well as effectively prevent the dramatic fluctuation of performance and the spread of detrimental information from individuals with inaccurate fitness values.

Our empirical studies in Section 5 show that SC can not only significantly reduce the computational cost of evolutionary optimization and boost the performance of original hybrid approaches, but also stabilize the interactions between the RL agent and the genetic population. To the best of our knowledge, this is the first attempt on introducing the surrogate model into the hybrid RL frameworks, and the major contributions of our work are summarized as follows:

- A novel module named Surrogate-assisted Controller (SC) is proposed that can be easily integrated into existing hybrid RL frameworks to significantly reduce the computational cost of the optimization process.
- Two effective management strategies are presented for SC to control the workflow of the hybrid frameworks, which can strategically schedule the surrogate-assisted evaluation and the real fitness evaluation.
- We combine SC with ERL and PDERL, creating two new frameworks named SERL and SPDERL, respectively, to highlight its principle and flexibility. Comprehensive experimental studies show that SC can not only effectively reduce the cost of fitness evaluations, but also stabilize the learning and evolution processes with superior performance.

The remainder of this paper is organized as follows. Section 2 introduces the related work on hybrid frameworks and surrogate-assisted methods for solving RL problems. The problem definition is specified in Section 3 and the details of SC and its components are presented in Section 4. In Section 5, the numerical validation and in-depth analyses are conducted and Section 6 concludes our work with some discussions on future research directions.

2. Related work

2.1. Hybrid frameworks based on EA and DRL

As an alternative solution for RL problems [11], EAs have also been combined with DRL to leverage the benefits of both solutions. The first hybrid framework ERL [12] combines an off-policy RL agent based on DDPG [3] with a population evolved by the Genetic Algorithm (GA) [29]. In each generation, individuals are evaluated over a few episodes and the fitness is given by averaging the total rewards. Based on this information, policy optimization is then conducted over the parameter space by genetic operators. Meanwhile, the RL agent is trained on the diverse experiences produced by the population. In the periodical synchronization step, it is injected into the population and this bi-directional interaction controls the information flow between the RL agent and the genetic population. To further extend the ERL framework, PDERL [15] uses the distillation crossover operator to alleviate the catastrophic forgetting caused by the recombination of neural networks. In addition, various evolutionary methods [16, 17, 18] have also been combined with other DRL frameworks such as TD3 [30] and SAC [31] to further leverage the advantages of both gradient-based and gradient-free methods. In our paper, we mainly focus on the paradigm of ERL and PDERL to show the effectiveness of our proposed methods.

2.2. Surrogate-assisted methods for RL problems

Surrogate-assisted methods have been widely investigated to reduce unnecessary expensive evaluations [32, 33]. Recently, a few studies on using the surrogate model to enhance RL algorithms or EAs in the context of sequential decision-making tasks have been conducted and can be generally divided into two categories.

The first category focuses on learning a model of the environment [34]. Many studies [35, 36, 37, 38] try to construct a transition model, which is trained in a standard supervised manner using a large amount of historical data, to imitate at least some aspects of the system's physical dynamics. Then, the fixed surrogate model is embedded into RL algorithms to help learn a control strategy from

the experiences obtained by continuous interactions with the surrogate model. However, for domains with high noise or when the availability of the historical data is limited, this kind of methods may no longer be effective.

The other category does not require modeling the environment and mainly focuses on evaluation. For example, Kriging can be employed to directly map the relationships between neural networks (policies) and their fitness. Typically, genotypic distances [39] between neural networks are needed by the Kriging model [40]. However, the large-scale settings and complex problems may make the computation of these distances practically impossible. Although approximate distances such as the phenotypic distance [27] can offer some help, the high dimensionality of the inputs can result in the increase of computational costs, and the parameter settings for Kriging remain a challenge for input vectors of different sizes [27].

For instance, Evolutionary Surrogate-assisted Prescription (ESP) [41] incorporates a surrogate model into the EA. Given a set of input states $\mathcal{S}$, the policy neural network takes each $s \in \mathcal{S}$ as the input and outputs the action a. The surrogate model of ESP, represented by a random forest or a deep neural network, is used to predict the outcome of each state-action pair (s, a), and the fitness of each policy is given by averaging the outcomes over $\mathcal{S}$. In each generation, the surrogate is first trained on the historical data by minimizing the Mean Square Error (MSE) loss between the real and the predicted outcomes. Subsequently, individuals (called *prescriptors*) in the population are evolved with the trained surrogate. Finally, selected elites are presented to the real environment to generate new training data for the surrogate. Unfortunately, for challenging environments with large continuous state and action spaces or with rich feedback signals, the lack of gradient information makes EAs suffer from brittle convergence [11, 12]. Furthermore, the update of the parameter in ESP is purely based on predicted fitness, which increases the risk of misleading the evolutionary optimization in the wrong direction under complex circumstances. Consequently, the frequency of applying the surrogate also needs to be managed properly to ensure the stability of the training process.

In general, apart from limited successes on simple discrete control tasks such as Cart-Pole [19], conventional surrogate-assisted EAs [27, 40, 41] still face significant challenges. To explore the potential of surrogate-assisted methods in continuous and complex RL contexts, in this work, we extend the surrogate model to hybrid RL frameworks with the objective to reduce the computational cost while making full use of the efficiency of the RL agent and the exploration capability of the EA.

3. Preliminaries

In DRL, each problem is modeled as a Markov Decision Process (MDP), which can be specified by a 5-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$. With the state space $\mathcal{S}$ and the action space $\mathcal{A}$, $\mathcal{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the transition function of the environment; $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and the discount factor $\gamma \in (0, 1]$ specifies the degree to which rewards are discounted over time.

$$Q(s, a|\theta^Q) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \Big| s_t = s, a_t = a\right] \tag{1}$$

The actor-critic architecture based on the policy gradient approach is widely used in DRL [5]. The critic network $Q(s, a|\theta^Q)$, as shown in Eq.(1), is used to estimate the expectation of the discounted accumulative rewards of the state-action pair (s, a), evaluating the actions produced by the actor. According to the Bellman equation, its recursive expression is:

$$Q(s, a|\theta^Q) = \mathbb{E}\left[r(s, a) + \gamma Q(s', a'|\theta^Q)\right] \tag{2}$$

where s' and a' represent the next state and action, respectively. In each iteration, transitions with batch size of N are sampled randomly from the experience replay buffer to update the parameters of $Q(s, a|\theta^Q)$ by minimizing the Temporal Difference (TD) loss based on the standard back-propagation:

$$\mathscr{L}_{Q(s,a|\theta^Q)} = \frac{1}{N} \sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2 \tag{3}$$

$$y_i = r(s_i, a_i) + \gamma Q\left(s', \mu(s'|\theta^\mu)|\theta^Q\right) \tag{4}$$

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i Q\left(s_i, \mu(s_i|\theta^\mu)\right) \nabla_{\theta^\mu} \mu(s_i|\theta^\mu) \tag{5}$$

Then, the critic is used to assist the training of the actor network $\mu(s|\theta^\mu)$ via policy gradient, according to Eq.(5). With the back-propagation method, the parameter θ^μ of the actor network is updated in the direction towards maximizing the Q value.

4. Methodology

In this section, we present the Surrogate-assisted Controller (SC) with two management strategies and introduce how to incorporate it into the hybrid framework. Figure 2 shows an example of combining SC with ERL, where the RL-Critic exploits the diverse experiences collected by the genetic population to simultaneously train the RL-Actor and assist the evaluation, referred to as Surrogate-assisted Evolutionary Reinforcement Learning (SERL). In practice, SC includes three components: critic-based surrogate model, management strategies and evaluation memory.

4.1. Critic-based surrogate model

Using real fitness functions for evaluation can be very time-consuming or even dangerous for expensive problems.

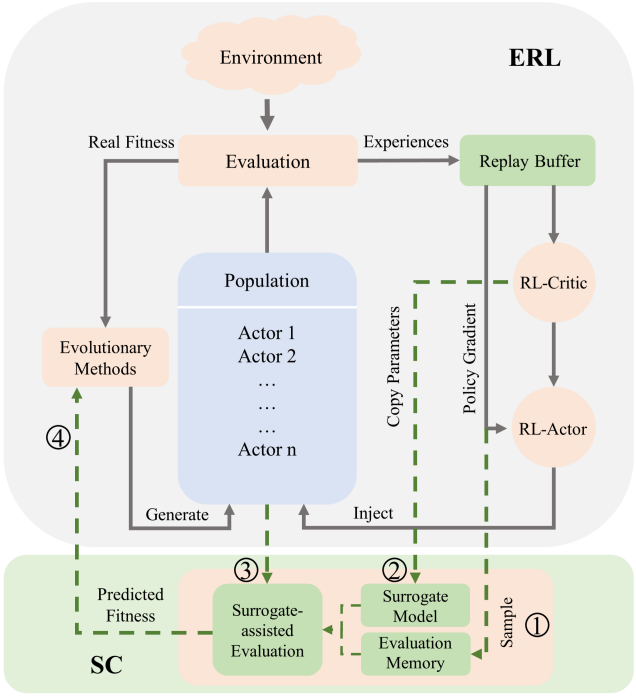


Figure 2: Integration of SC and ERL. The ERL framework is shown in the light gray box and SC is shown in the light green box. To evaluate the population without environment interactions, the recent state information is sampled from the replay buffer as the evaluation memory. After that, the RL-Critic plays the role of a surrogate model to evaluate actors based on the historical data. Finally, the predicted population fitness is consumed by evolutionary methods.

Algorithm 1 Surrogate-assisted Evaluation

Input: Policy set of population $\mu_{pop} = \{\mu_1, \mu_2, ..., \mu_n\}$;
The surrogate model $Q(s, a|\theta^Q)$;
Replay buffer $\mathcal{R}$ for RL agent training.
Output: Population fitness F

Sample k latest states from $\mathcal{R}$ to the evaluation memory $\mathcal{R}_{eva}$ as the evaluation samples;
for $i = 1$ to n **do**
  Initialize the fitness of μ_i: $f_i = 0$;
  for $j = 1$ to k **do**
    $f_i = f_i + Q\big(s_j, \mu_i(s_j|\theta^{\mu_i})|\theta^Q\big)/k$;
  end for
end for
return Population fitness $F = \{f_1, f_2, ..., f_n\}$.

A surrogate, which is a predictive model, can be used to partially replace the original fitness function. For sequential decision-making problems, the genotype-phenotype-fitness mapping is often difficult to learn but it is straightforward for the surrogate model to estimate the outcome of a state-action pair (s, a). In ESP [41], individuals are evaluated by an extra dedicated surrogate model. However, in hybrid RL frameworks, the critic $Q(s, a|\theta^Q)$ of the RL agent can be naturally used as a surrogate. Thus, given the information of

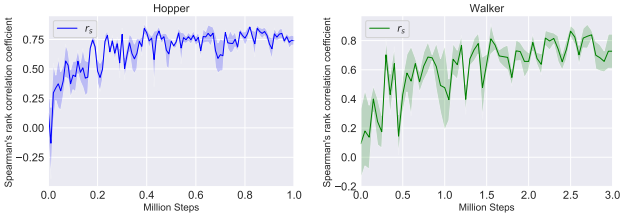


Figure 3: The preliminary experiment on the surrogate's approximation accuracy over two environments: Hopper (left, trained for 1 million steps) and Walker (right, trained for 3 million steps). The curves of approximation accuracy clearly show that the accuracy of the surrogate tends to increase during the training process.

k states, the fitness value of a policy neural network $\mu(s|\theta^\mu)$ from the genetic population can be obtained by averaging its predicted Q values over all states, as shown in Eq.(6). The complete process of the surrogate-assisted evaluation is presented in Algorithm 1.

$$f = \frac{1}{k}\sum_{j=1}^{k} Q\big(s_j, \mu(s_j|\theta^\mu)|\theta^Q\big) \tag{6}$$

A prominent feature of our method is that there is no extra cost involved in training the surrogate model, as it is part of the standard training procedure of the RL. With various improvements in the estimation of Q values [30, 42], the critic-based surrogate model can simultaneously train the RL-Actor via policy gradient and provide relatively accurate fitness estimations for individuals.

The approximation quality of the surrogate is measured by the Spearman's rank correlation coefficient $r_s \in [-1, 1]$ between the real and the predicted fitness values, where n is the number of data points (population size) and d represents the difference between the two ranks, as shown in Eq.(7). A preliminary experiment on two standard continuous control tasks from MuJoCo [43] is conducted to show how the approximation accuracy of the critic-based surrogate changes during ERL's training process. In each generation, we apply the surrogate-assisted evaluation on the newly generated population with $k = 50,000$ in the first place. After evaluating the population in the real environment, r_s is calculated to report the current approximation accuracy of the surrogate.

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{7}$$

The results in Figure 3 suggest that, at the beginning of training, the surrogate has relatively low approximation accuracy. However, previous studies [32, 41, 44] suggest that this kind of uncertainty may not cause negative effects for evolutionary search. Instead, it may further push the population to explore the fitness landscape. Moreover, the training data generated by these candidates can subsequently help improve the approximation accuracy of the surrogate.

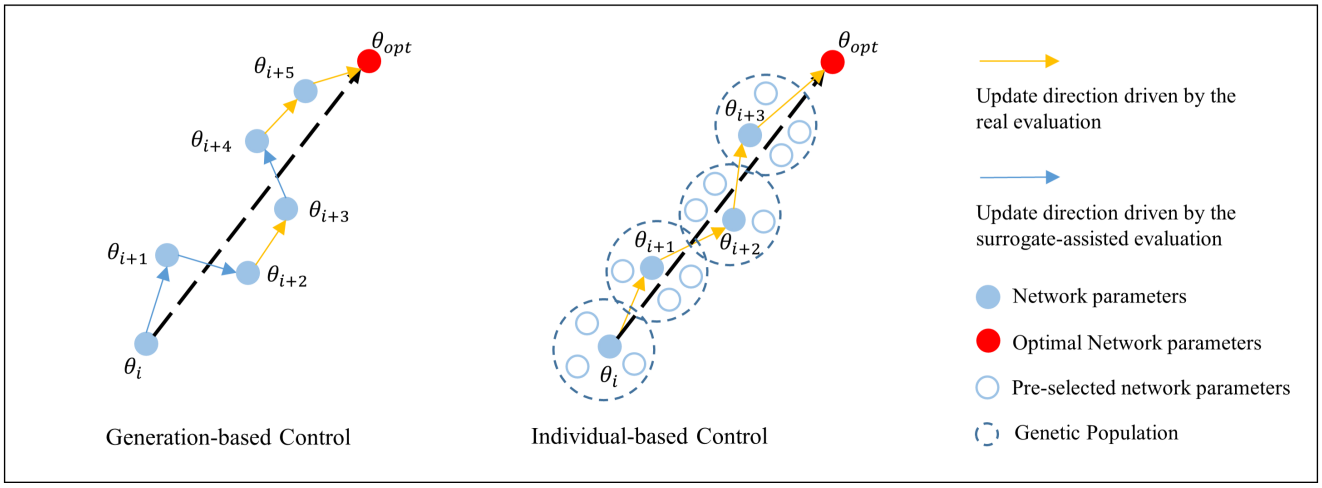


Figure 4: Two kinds of management strategies: the Generation-based Control (GC) and the Individual-based Control (IC). For GC, the parameter update direction is partially based on real fitness information, while the update direction is totally provided by the real fitness of the pre-selected candidates in IC.

Algorithm 2 Generation-based control

Input: Policy set of population $\mu_{pop} = \{\mu_1, \mu_2, ..., \mu_n\}$;
The surrogate model Q_{rl};
Replay buffer $\mathcal{R}$ for RL agent training;
Control factor ω;
Random number generator $G_{rand} \in (0, 1]$.
Output: New population μ_{pop^*}

if $G_{rand} > \omega$ **then**
  F_{real} = Evaluation(μ_{pop})
  Copy the elite actor
  μ_{pop^*} = Evolutionary methods(F_{real}, μ_{pop})
else
  F_{pre} = Surrogate-assisted evaluation(μ_{pop}, Q_{rl}, $\mathcal{R}$)
  μ_{pop^*} = Evolutionary methods(F_{pre}, μ_{pop})
  Maintain the recorded elite actor in μ_{pop^*}
end if
return New population μ_{pop^*}

Algorithm 3 Individual-based control

Input: Policy set of population $\mu_{pop} = \{\mu_1, \mu_2, ..., \mu_n\}$;
The surrogate model Q_{rl}; The RL actor μ_{rl};
Replay buffer $\mathcal{R}$ for RL agent training;
Candidate population size n^*; Scaling factor σ.
Output: New population μ_{pop^*}

for $i = 1$ to $n^* - n$ **do**
  Sample $\epsilon_i \sim \mathcal{N}(0, 1)$
  Inject a new candidate $(\mu_{rl} + \sigma\epsilon_i)$ to μ_{pop}
end for
F_{pre} = Surrogate-assisted evaluation(μ_{pop}, Q_{rl}, $\mathcal{R}$)
Retain the best $n - 1$ actors in μ_{pop} according to F_{pre}
Inject the recorded elite actor to μ_{pop}
F_{real} = Evaluation(μ_{pop})
Copy the elite actor
μ_{pop^*} = Evolutionary methods(F_{real}, μ_{pop})
return New population μ_{pop^*}

4.2. Management strategies

Although the surrogate model can provide an approximately accurate fitness estimation, using predicted fitness to assist the evolutionary operations throughout the optimization process may easily introduce false minima, leading to a drop in performance or an increase in the computational cost. As a result, surrogates should be used along with real fitness functions and we consider the following two strategies: generation-based and individual-based strategies.

4.2.1. Generation-based control

In the generation-based control, a fixed hyperparameter $\omega \in [0, 1)$ is used to indicate the probability of using the surrogate-assisted evaluation. In this setting, the evolutionary operators are partially based on predicted fitness values. Note that, at the beginning of evolution, the population is evaluated in the real environment to collect necessary state data. Algorithm 2 shows the pseudo-code of the generation-based control.

Previous studies have provided some theoretical analyses of the convergence of surrogate-assisted EAs [45, 46]. Here, we demonstrate how the parameter update path is affected by different management strategies. In Figure 4, although inaccuracy may be introduced by the surrogate model under the generation-based control, misguiding the update direction, it can be corrected appropriately based on real fitness evaluations. Thus, the update path may be oscillatory as SC switches between the real and the predicted fitness landscapes.

4.2.2. Individual-based control

In our individual-based approach, the evolutionary operators work on real fitness. Assume that the population size is n. Before the population is evaluated in the real environment,

a candidate population with n^* offspring is generated with $n^* > n$. After being evaluated by the surrogate model, only the best n individuals are presented to the real environment. This method is also referred to as "preselection strategy". In principle, to generate the candidate population, apart from the original population, $n^* - n$ extra individuals are produced by adding Gaussian noise to the RL actor or the best actor found so far (Algorithm 3). By default, we mutate the RL actor to better explore its surrounding landscape.

As shown in Figure 4, the surrogate model preselects those mutated individuals with relatively higher predicted fitness, and the preselected genetic population actually forms a "trust region" [47]. The real fitness evaluation is then conducted for more stable optimization, resulting in a more directional and smooth parameter update path.

4.2.3. Elite protection

An important issue that needs to be addressed is to protect elites from being discarded if the surrogate model is used for fitness evaluation. For this purpose, SC keeps track of the current best actor when the population is evaluated by the real fitness function and maintains it in the population after performing evolutionary operators based on predicted fitness. This operation effectively prevents the dramatic fluctuation of performance and the spread of detrimental information from individuals with inaccurate fitness while using the fixed evolution control.

4.3. Evaluation memory

Off-policy RL methods such as DQN [6], DDPG [3], and TD3 [30] maintain a constantly updated replay buffer to improve the sample efficiency of the RL agent. In our approach, the most recent part of these historical data (state information only), referred to as evaluation memory, is exploited to evaluate the population. This memory not only contains diverse state samples but also keeps track of the current optimization process. In addition, evaluation memory does not have to be maintained all the time. It is only created when the surrogate model is called for evaluating individuals and has a negligible memory footprint.

5. Experiments and Evaluations

To highlight the value of SC, we focus on the implementation of SC with two state-of-the-art hybrid frameworks ERL and PDERL, referred to as SERL and SPDERL, respectively. Our experiments contain six continuous control tasks from MuJoCo environments [43] and aim to answer the following questions: (1) Does SC improve the computational efficiency and the performance of the original hybrid frameworks? (2) How sensitive is the optimization process to the control parameters of SC's management strategies? (3) What impacts does SC bring to the internal dynamics of the original hybrid frameworks?

5.1. Experimental setting

We use the official implementations of ERL[1] [12] and PDERL[2] [15] as the major baselines and follow all their hyperparameter settings for EAs, RL agents, neural networks and the population size ($n = 10$). The periodical synchronization of the RL actor and the population is performed only after real fitness evaluation. Furthermore, we use a standard Genetic Algorithm [12] and the DDPG implemented by OpenAI Spinningup [3] as extra baselines to compare our methods against pure EA and RL algorithms. More details on parameter settings are listed in Appendix A.

For SC, the maximum evaluation memory size k is limited to $50,000$, and we fix ω to 0.6 for the generation-based control, which means that, in each generation, the population has a 60% chance of being evaluated by the surrogate model. For the individual-based control, $\alpha = (n^* - n)/n$ is the control factor for the candidate population size with $\alpha = 1$ in our experiments and we set the scaling factor $\sigma = 0.01$ for generating Gaussian noise. During the training, SC keeps recording the current best actor when the population is evaluated in the real environment for elite protection.

We refer to SERL and SPDERL with generation-based control as SERL-G and SPDERL-G, respectively. Under this control strategy, the current best actor recorded by SC will replace the individual with the lowest predicted value if the evolutionary operators are performed based on predicted fitness. Similarly, SERL-I and SPDERL-I indicate the combination of SERL and PDERL with individual-based control, respectively. Before the real fitness evaluation, the $n^* - n$ actors are generated by adding Gaussian noise to the current RL actor and after surrogate-assisted evaluation, the top $n-1$ actors from the candidate population and the current best actor construct a new population to be applied to the real environment.

5.2. Overall performance

To answer the first research question, we train each proposed method with six different random seeds on six MuJoCo environments following the convention in literature [43]. Figure 5 illustrates their learning processes during training. The solid curves represent the mean values and the shaded regions indicate the standard deviations.

In most environments, SC can significantly improve the learning speed and the performance of the original hybrid frameworks and also make the learning process more stable with lower variance. For example, SERL-G can outperform ERL across all environments. Except for Swimmer, the improvement of SERL-I is more evident in the early training phase (within 1 million steps) compared to SERL-G, and it outperforms other methods in Reacher. When it comes to SPDERL, both SPDERL-I and SPDERL-G outperform PDERL in Hopper and Walker. For Ant and HalfCheetah, SPDERL-I can achieve higher final performance while SPDERL-G can accelerate the performance improvement

[1] https://github.com/ShawK91/Evolutionary-Reinforcement-Learning
[2] https://github.com/crisbodnar/pderl
[3] https://github.com/openai/spinningup

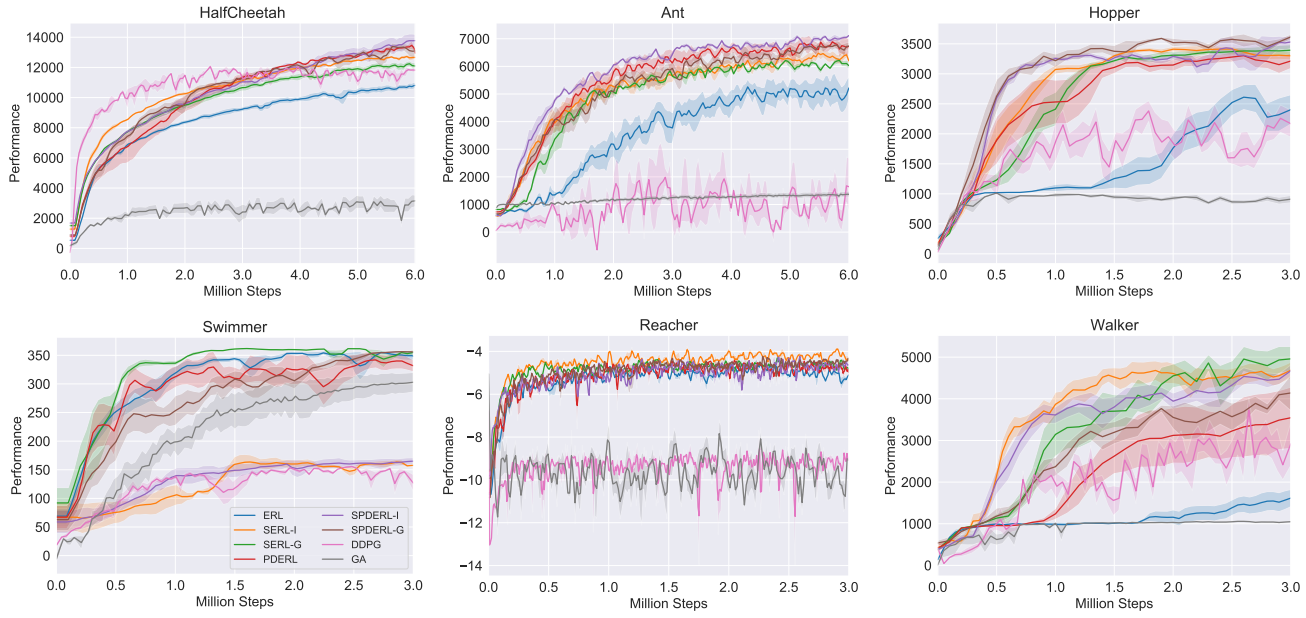


Figure 5: Learning curves on six MuJoCo environments: HalfCheetah, Ant, Hopper, Swimmer, Reacher and Walker.

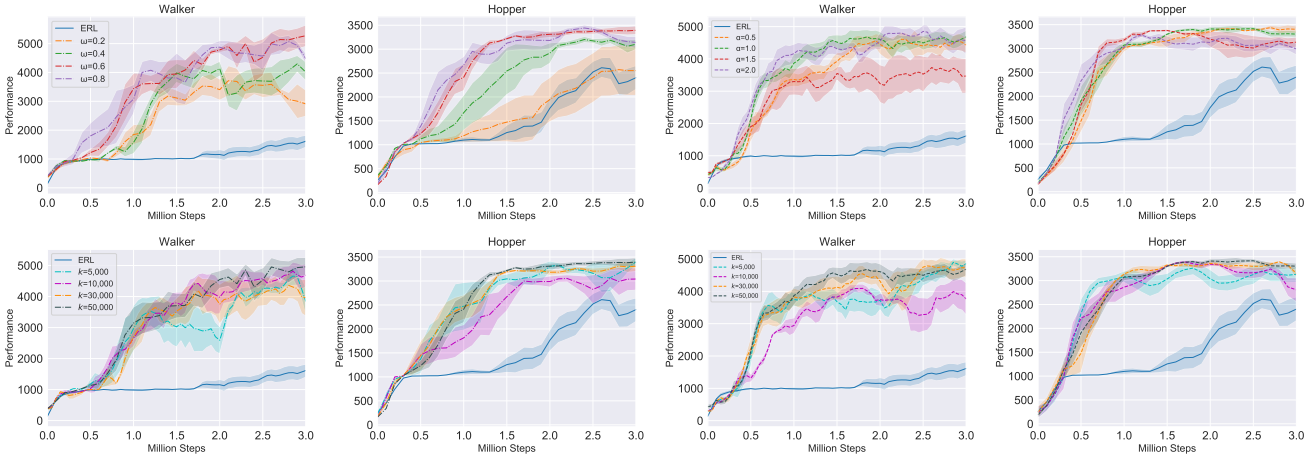


Figure 6: Parameter analysis of SERL-G with different generation-based control factors and evaluation memory sizes (first two columns) and SERL-I with different individual-based control factors and evaluation memory sizes (last two columns) in Walker and Hopper environments.

in the early training phase. Furthermore, we also combine SC with another state-of-the-art hybrid framework CEM-RL [16] to further highlight its effectiveness and the additional experimental results can be found in Appendix B.

It is worth noting that all methods under the individual-based control fail in Swimmer and, as explained in [12, 16], DRL methods face great challenges in effectively learning the gradient information. To alleviate this issue, we generate the candidate population by mutating the best actor that has been found by genetic operations (Appendix C). In this specific environment, the evolutionary search is more suitable for driving the optimization process.

5.3. Parameter analysis

In this part, we investigate the influence of the following parameters: the ratio of surrogate-assisted evaluation ω, the control factor α of the candidate population size, and the capacity k of evaluation memory, as shown in Figure 6.

Control factors. We vary ω from 0.2 to 0.8 in SERL-G and α from 0.5 to 2.0 in SERL-I, respectively. The results indicate that, for the generation-based method, both the final performance and the learning speed are generally improved by increasing ω. However, an excessively large value (e.g., $\omega = 0.8$) may result in fluctuation and undesirable performance. In practice, the recommended range of ω is from 0.4

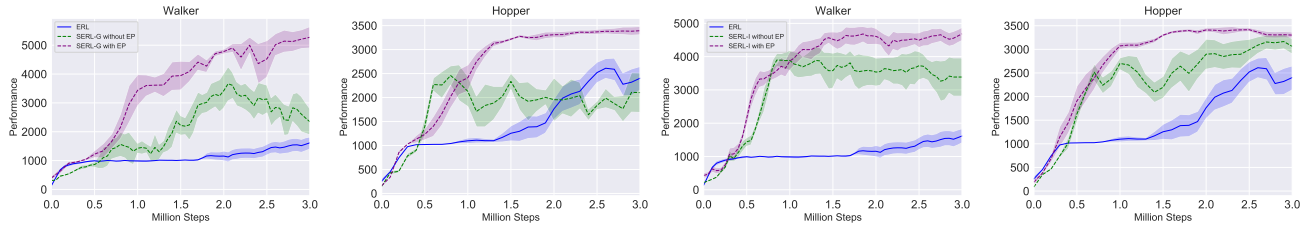


Figure 7: Comparisons of SERL-G (first two figures) and SERL-I (last two figures) with and without Elite Protection (EP) mechanism in Walker and Hopper environments.

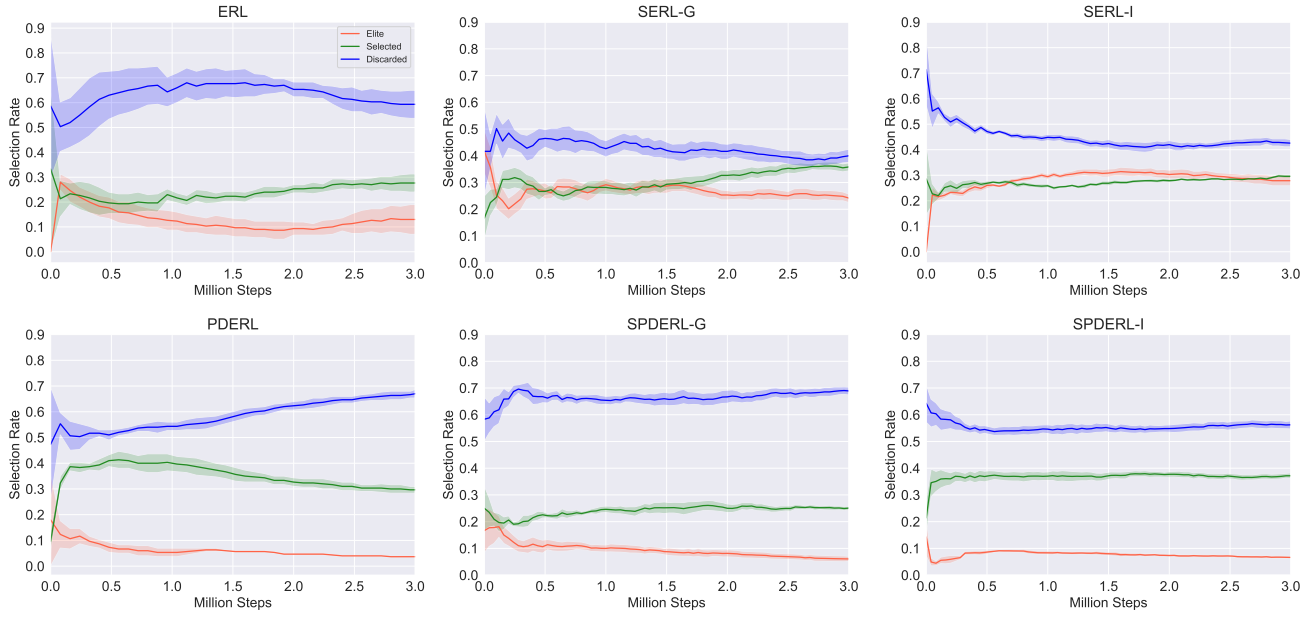


Figure 8: Accumulative rates of the RL agent being selected, discarded, or chosen as the elite during the training process in Hopper. The results indicate that SC can potentially stabilize the internal dynamics of the original frameworks.

to 0.6. In addition, a relatively small α value is more cost-effective and its recommended range is from 0.5 to 1.0.

Memory capacity. Furthermore, we investigate the impact of the capacity of the evaluation memory. The second row of Figure 6 shows the performance of SERL-G and SERL-I with fixed control factors $\omega = 0.6$, $\alpha = 1.0$ and various k values in two environments. Overall, a large evaluation memory contains more diverse state data and significantly helps improve the quality of evaluation, but it also increases the computational cost. In general, the generation-based control is better suited with large evaluation memories to counteract the bias in surrogate-assisted evaluation. By contrast, since the evolutionary methods are based on real fitness evaluation with low deviations, a relatively small k is suitable for individual-based control.

5.4. Elite protection evaluation

Figure 7 shows the performance of SERL-I and SERL-G without the elite protection mechanism. In this setting, SC only speeds up the policy improvement in the early period of the training process and then encounters a dramatic drop in performance. Although the surrogate model can provide a roughly accurate estimation of population fitness, its estimation of elites is possibly biased, which increases the risk of discarding elites from the population. This experiment underlines the importance of elite protection while using the surrogate for fitness evaluation.

5.5. Changes of the internal dynamics

To gain a deeper insight into the internal transformation of hybrid frameworks in the presence of SC, we highlight the changes in the internal dynamic between the RL agent and the genetic population. More importantly, we experimentally explore how SC influences the update path of the policy's parameter under different management strategies.

Interactions between RL and EA. We keep a separate record of the accumulative rates of the RL actor being selected, discarded, or chosen as an elite in the population in Hopper. In Figure 8, although ERL, SERL-G, and SERL-I present similar dynamics, the integration of SC significantly stabilizes the internal dynamic of learning and evolution, which is more pronounced in PDERL and the two variants of SPDERL, where the evolutionary search is the major driving force for the training process. It is reasonable to hypothesize

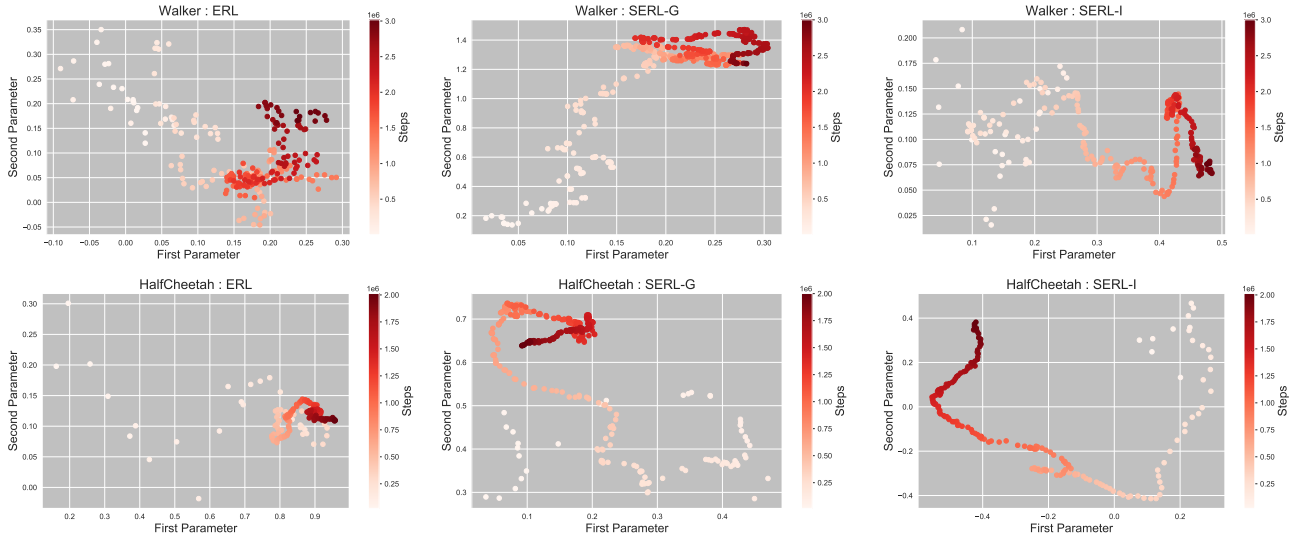
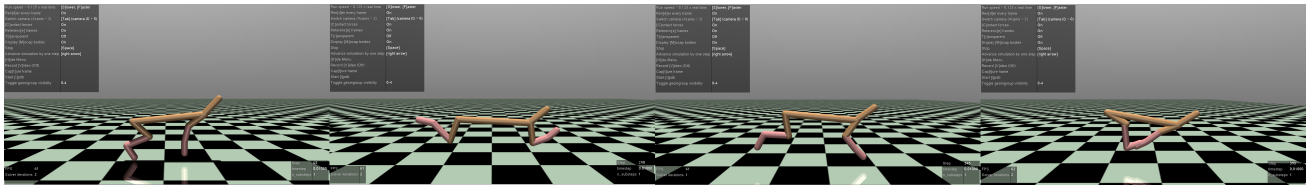
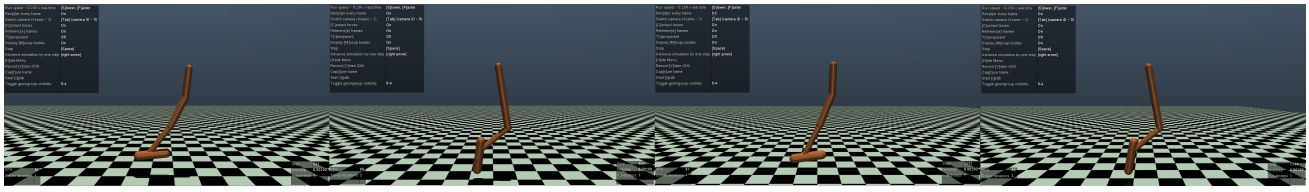


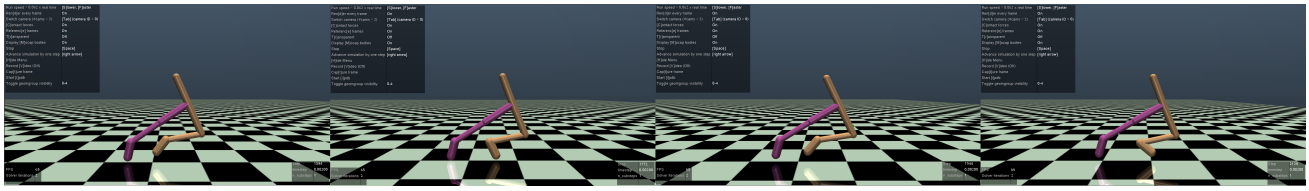
Figure 9: Update paths of the first two parameters of the RL policy network during the training of ERL, SERL-G and SERL-I on Walker and HalfCheetah environments.



(a) HalfCheetach agent



(b) Hopper agent



(c) Walker agent

Figure 10: Several intriguing behavioral patterns of the agents trained by SPDERL-I and SPDERL-G. (a) A HalfCheetah agent trained by SPDERL-I with average performance of 14000 points over 50 test seeds. The agent is able to adjust its posture more appropriately and run faster. (b) A Hopper agent trained by SPDERL-I with average performance of 4100 points over 50 test seeds. The agent jumps faster and learns to better stabilize the center of gravity. (c) A Walker agent trained by SPDERL-G with average performance of 9000 points over 50 test seeds. The agent learns to walk with only one leg and use the other leg to maintain balance.

that the new fitness landscapes constructed by the surrogate may provide better guidance for the evolutionary search and make the optimization process more stable.

Parameter update path. In ERL and PDERL, the RL agent indirectly benefits from all the experiences collected by the genetic population, and these experiences would greatly force the RL agent to perform gradient steps. Figure 9 illustrates the optimization processes of the first two parameters of the RL agent's policy network during the training of ERL, SERL-G and SERL-I on Walker and HalfCheetah and

Table 1
The average time steps (in million, equal to the number of consumed samples) of different algorithms in various environments when reaching the target score. "–" represents that DDPG or GA can not reach the target.

Env	Score	DDPG	GA	ERL	SERL-G	SERL-I	PDERL	SPDERL-G	SPDERL-I
Ant	5,000	–	–	4.266	**1.564**	**1.685**	1.464	1.784	**1.085**
Hopper	2,500	–	–	2.519	**1.073**	**0.786**	0.857	**0.554**	**0.561**
Walker	2,000	0.775	–	3.572	**0.767**	**0.505**	1.312	**0.796**	**0.508**
Swimmer	300	–	2.640	0.968	**0.516**	1.034	0.742	1.489	3.568
Reacher	−5	–	–	1.051	**0.322**	**0.224**	0.717	**0.513**	**0.672**
HalfCheetah	10,000	0.986	–	4.092	**2.565**	**1.761**	2.234	**2.153**	**2.201**

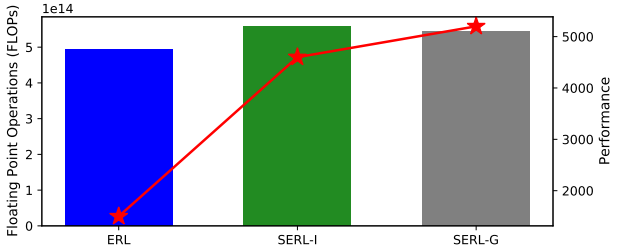


Figure 11: Comparison of ERL, SERL-I and SERL-G in terms of the number of FLOPs and performance (red star) achieved at 3M environment steps in Walker.

the color of the dots gets darker as the training proceeds. For ERL on Walker, the parameter space tends to be explored in a more random manner, leading to poor performance (Figure 5). Meanwhile, on HalfCheetah, the optimization process of ERL is prematurely trapped in the local optimum, significantly slowing down the improvement of policy. By contrast, both SERL-G and SERL-I converge to high-quality solutions by effectively exploring the parameter space. Under the generation-based control, the update path exhibits an oscillatory property, and is more directional and stable under the individual-based control strategy, as we discussed in Section 4.2.

Intriguing behavioral patterns. A notable discovery is that the agents trained by our proposed methods can produce highly intriguing behavioral patterns than the original hybrid approaches, as shown in Figure 10. The solutions found by hybrid frameworks with SC are more intriguing and stable. On some control tasks, they can better adapt themselves to the environments and perform more competently.

5.6. Computational efficiency

Floating Point Operations. We show that SC is a practical and scalable module for different hybrid frameworks. The computational efficiency of SC is verified by comparing the Floating Point Operations (FLOPs) consumed by ERL, SERL-I, SERL-G and their corresponding performance (Figure 11). Regardless of which hybrid framework that SC is combined with, the computational cost of the entire training process can always be divided into the surrogate-assisted evaluation cost and the original optimization cost, as shown in Appendix D. In particular, while bringing significant performance improvement, SC is computationally efficient in that only a small amount of forward propagation for surrogate-assisted evaluations is introduced, and the computational time of these calculations is much less than real evaluation.

Sample consumption. Furthermore, we conduct a sample reduction study of different methods in various environments when reaching the target scores. Table 1 shows that SC can significantly reduce the sample consumption across most domains. For different control strategies, the individual-based control requires relatively less number of samples compared with generation-based control.

6. Conclusion and Future Work

The application of hybrid RL frameworks to expensive learning problems has largely been limited by the cost of evaluating the population in real environments. In this work, we propose a surrogate-assisted controller with two management strategies, which can be easily integrated into existing hybrid frameworks to simultaneously facilitate the optimization of the RL agent and the evaluation of the population. To the best of our knowledge, this is the first attempt on introducing the surrogate model into hybrid RL frameworks. Empirical evaluations show that the combination of SC with two state-of-the-art evolutionary reinforcement learning frameworks ERL and PDERL can effectively reduce the evaluation cost and boost the performance. Furthermore, SC brings beneficial changes to the internal dynamics of learning and evolution, resulting in more collaborative interactions between the RL agent and the EA population.

Learning and evolution are two symbiotic counterparts in nature. It is of great scientific importance to fully appreciate and explore this hybrid paradigm towards implementing truly competent artificial intelligence. As to future work, there are plenty of fascinating directions to advance our proposed techniques, including real-world settings, such as embodied AI, designing self-adaptive evolution control strategies and more effective evaluation criteria for the surrogate. For complex and challenging settings, multi-agent evolutionary reinforcement learning with multiple surrogates is expected to further extend the horizon.

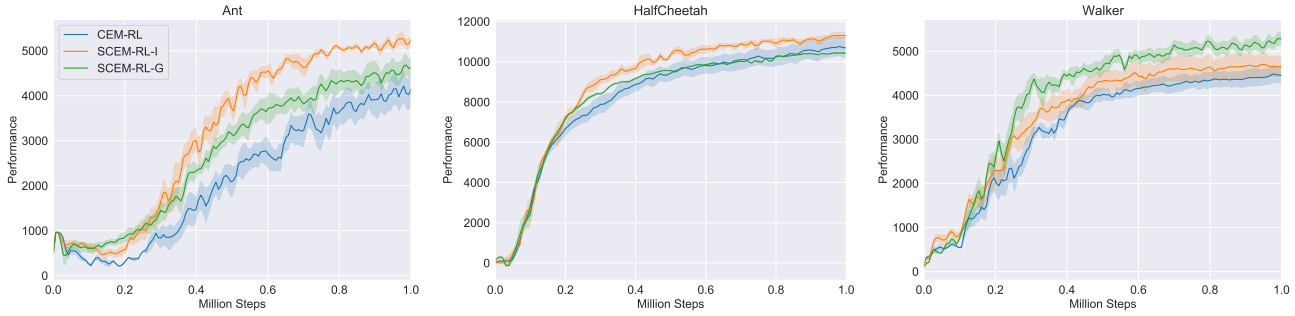


Figure 12: Learning curves on three MuJoCo environments: Ant, HalfCheetah and Walker.

Table 2
Hyperparameters of ERL and PDERL

Hyperparameter	Value
Target weight τ	0.001
RL Actor learning rate	$5e^{-5}$
RL Critic learning rate	$5e^{-4}$
Genetic Actor learning rate	$1e^{-3}$
Replay buffer size	$1e^{6}$
Genetic memory size	8000
RL Agent batch size	128
Discount factor	0.99
Genetic Agent crossover batch size	128
Genetic Agent mutation batch size	256
Distillation crossover epochs	12
Mutation probability	0.9

Table 3
Hyperparameters used in various environments

Environment	Algorithm	Elite	Trials	Sync
Ant	SERL	0.3	1	1
	SPDERL	0.2	1	1
Hopper	SERL	0.3	5	1
	SPDERL	0.2	3	1
Walker	SERL	0.2	3	1
	SPDERL	0.2	5	1
Swimmer	SERL	0.1	1	10
	SPDERL	0.1	1	10
Reacher	SERL	0.1	1	10
	SPDERL	0.1	1	10
HalfCheetah	SERL	0.1	1	1
	SPDERL	0.1	1	10

Appendix A. Hyperparameters

Table 2 shows the hyperparameters of ERL and PDERL in this work and Table 3 shows the settings used in various environments.

Appendix B. Combining SC with CEM-RL

CEM-RL is another state-of-the-art hybrid framework [16], which combines Cross-Entropy Method (CEM) with TD3 learners [30], an off-policy DRL algorithm related to DDPG [3]. We combine CEM-RL with our proposed SC, referred to as Surrogate-assisted CEM-RL (SCEM-RL), to further highlight its efficiency and flexibility.

We train SCEM-RL-G (SCEM-RL with generation-based control) and SCEM-RL-I (SCEM-RL with individual-based control) with six different random seeds. Figure 12 illustrates their learning curves during training on three control tasks from MuJoCo [43]. Apart from the clear advantages of SC in the training performance and sample consumption, similar to the results in Section 5.2, on HalfCheetah and Ant, the improvement over the original hybrid framework is more evident under the individual-based control, while the generation-based control method is more favorable on Walker.

Appendix C. The Swimmer environment

Results in Figure 5 demonstrate that all individual-based control methods fail in the Swimmer environment because the RL agent is unfortunately misled by the deceptive gradient information, and candidates mutated from the RL actor could not provide useful information for policy improvement. To alleviate this issue, we generate the candidate population by mutating the best actor that has been found by genetic operations (Figure 13). In this specific environment, the evolutionary search is more suitable for driving the optimization process.

Appendix D. Measurement of FLOPs

In this section, we introduce how the FLOPs values in Figure 11 are calculated. The neural networks in our work are all fully connected and the number of updates of the RL agent is equal to the environment steps. Similar to [48], we consider the FLOPs of each forward pass being half of that of the backward pass. We also assume that the consumed FLOPs of activation functions and operations that do not require passing through neural networks are negligible.

We follow the procedures in [49] to measure the FLOPs consumed by each method within 3M environment steps. The calculation formulas for FLOPs consumed by ERL, SERL-I and SERL-G are shown in Eqs.(8), (9) and (10), respectively.

$$F_{ERL} = T(A_f + b \times (2A_f + 3C_f + C_b + A_b)) \quad (8)$$

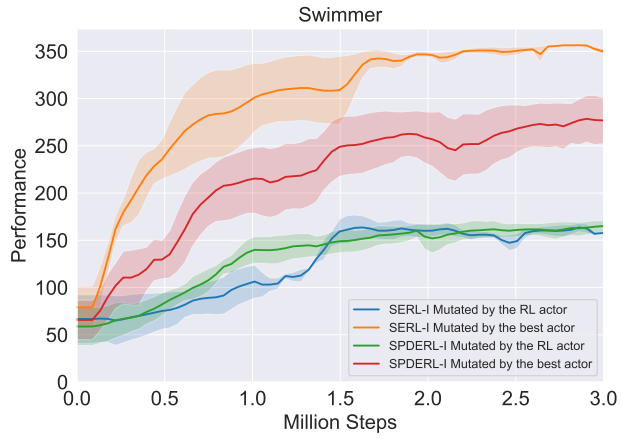


Figure 13: Learning curves using different approaches of generating the candidate population in the Swimmer environment.

Table 4
Symbolic meanings and values in the Calculation of FLOPs

Symbol	Value
Environment steps T	$3,000,000$
Population size n	10
Candidate population size n_c	20
RL Agent batch size b	128
Evaluation memory size k	$50,000$
Evolutionary generations G	240
Evolutionary generations using the surrogate G_s	390
Flops of forward propagation of RL-Actor A_f	$11,136$
Flops of forward propagation of RL-Critic C_f	$249,800$
Flops of backward propagation of RL-Actor A_b	$22,272$
Flops of backward propagation of RL-Critic C_b	$499,600$

$$F_{SERL(I)} = F_{ERL} + G_s \times n \times k \times (A_f + C_f) \quad (9)$$

$$F_{SERL(G)} = F_{ERL} + G \times n_c \times k \times (A_f + C_f) \quad (10)$$

All the symbols and their symbolic meanings that are involved in calculating are shown in Table 4.

References

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

[2] D. Silver, 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 529 (7587) (2016) 484–489.

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

[4] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[5] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602 (2013).

[7] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F. E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.

[8] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, A. Madry, A closer look at deep policy gradients, arXiv preprint arXiv:1811.02553 (2018).

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

[10] M. M. Drugan, Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms, Swarm and evolutionary computation 44 (2019) 228–246.

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

[12] S. Khadka, K. Tumer, Evolution-guided policy gradient in reinforcement learning, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 1196–1208.

[13] B. H. Weber, D. J. Depew, Evolution and learning: The Baldwin effect reconsidered, Mit Press, 2003.

[14] A. Gupta, S. Savarese, S. Ganguli, L. Fei-Fei, Embodied intelligence via learning and evolution, arXiv preprint arXiv:2102.02202 (2021).

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

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

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

[18] K. Suri, X. Q. Shi, K. N. Plataniotis, Y. A. Lawryshyn, Maximum mutation reinforcement learning for scalable control, arXiv preprint arXiv:2007.13690 (2020).

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

[20] G. Schneider, Neural networks are useful tools for drug design, Neural Networks 13 (1) (2000) 15–16.

[21] Y. Jin, M. Olhofer, B. Sendhoff, Managing approximate models in evolutionary aerodynamic design optimization, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), Vol. 1, IEEE, 2001, pp. 592–599.

[22] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft computing 9 (1) (2005) 3–12.

[23] M. ZAEFFERER, Surrogate models for discrete optimization problems (2018).

[24] A. Keane, A. Forrester, A. Sobester, Engineering design via surrogate modelling: a practical guide, American Institute of Aeronautics and Astronautics, Inc., 2008.

[25] C. Audet, J. Denni, D. Moore, A. Booker, P. Frank, A surrogate-model-based method for constrained optimization, in: 8th symposium on multidisciplinary analysis and optimization, 2000, p. 4891.

[26] I. Loshchilov, M. Schoenauer, M. Sebag, A mono surrogate for multiobjective optimization, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, 2010, pp. 471–478.

[27] J. Stork, M. Zaefferer, T. Bartz-Beielstein, A. Eiben, Surrogate models for enhancing the efficiency of neuroevolution in reinforcement learning, in: Proceedings of the genetic and evolutionary computation conference, 2019, pp. 934–942.

[28] A. Ratle, Optimal sampling strategies for learning a fitness model, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Vol. 3, IEEE, 1999, pp. 2078–2085.

[29] D. Whitley, A genetic algorithm tutorial, Statistics and computing 4 (2) (1994) 65–85.

[30] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.
[31] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, arXiv preprint arXiv:1812.05905 (2018).
[32] Y. Jin, Surrogate-assisted evolutionary computation: Recent advances and future challenges, Swarm and Evolutionary Computation 1 (2) (2011) 61–70.
[33] B. Liu, Q. Zhang, G. G. Gielen, A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, IEEE Transactions on Evolutionary Computation 18 (2) (2013) 180–192.
[34] R. Koppejan, S. Whiteson, Neuroevolutionary reinforcement learning for generalized control of simulated helicopters, Evolutionary intelligence 4 (4) (2011) 219–241.
[35] D. Cao, J. Zhao, W. Hu, F. Ding, N. Yu, Q. Huang, Z. Chen, Model-free voltage control of active distribution system with pvs using surrogate model-based deep reinforcement learning, Applied Energy 306 (2022) 117982.
[36] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, arXiv preprint arXiv:1809.01999 (2018).
[37] N. Wahlström, T. B. Schön, M. P. Deisenroth, From pixels to torques: Policy learning with deep dynamical models, arXiv preprint arXiv:1502.02251 (2015).
[38] M. Deisenroth, C. E. Rasmussen, Pilco: A model-based and data-efficient approach to policy search, in: Proceedings of the 28th International Conference on machine learning (ICML-11), Citeseer, 2011, pp. 465–472.
[39] J. Stork, M. Zaefferer, A. Fischbach, F. Rehbach, T. Bartz-Beielstein, Surrogate-assisted learning of neural networks (2017).
[40] J. Stork, M. Zaefferer, T. Bartz-Beielstein, Improving neuroevolution efficiency by surrogate model-based optimization with phenotypic distance kernels, in: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), Springer, 2019, pp. 504–519.
[41] O. Francon, S. Gonzalez, B. Hodjat, E. Meyerson, R. Miikkulainen, X. Qiu, H. Shahrzad, Effective reinforcement learning through evolutionary surrogate-assisted prescription, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, 2020, pp. 814–822.
[42] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 30, 2016.
[43] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.
[44] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, K. Giannakoglou, Metamodel—assisted evolution strategies, in: International Conference on parallel problem solving from nature, Springer, 2002, pp. 361–370.
[45] Y. Chen, W. Xie, X. Zou, How can surrogates influence the convergence of evolutionary algorithms?, Swarm and Evolutionary Computation 12 (2013) 18–23.
[46] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. K. Tucker, Surrogate-based analysis and optimization, Progress in aerospace sciences 41 (1) (2005) 1–28.
[47] M. Powell, On the convergence of a wide range of trust region methods for unconstrained optimization, IMA journal of numerical analysis 30 (1) (2010) 289–301.
[48] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, K. Lee, State entropy maximization with random encoders for efficient exploration, arXiv preprint arXiv:2102.09430 (2021).
[49] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440 (2016).