Two-Stage Evolutionary Reinforcement Learning for Enhancing Exploration and Exploitation

Qingling Zhu[1], Xiaoqiang Wu[2], Qiuzhen Lin[2*], Wei-Neng Chen[3]

[1]National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen, China
[2]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[3]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
qiuzhlin@szu.edu.cn

Abstract

The integration of Evolutionary Algorithm (EA) and Reinforcement Learning (RL) has emerged as a promising approach for tackling some challenges in RL, such as sparse rewards, lack of exploration, and brittle convergence properties. However, existing methods often employ actor networks as individuals of EA, which may constrain their exploratory capabilities, as the entire actor population will stop evolving when the critic network in RL falls into local optima. To alleviate this issue, this paper introduces a Two-stage Evolutionary Reinforcement Learning (TERL) framework that maintains a population containing both actor and critic networks. TERL divides the learning process into two stages. In the initial stage, individuals independently learn actor-critic networks, which are optimized alternatively by RL and Particle Swarm Optimization (PSO). This dual optimization fosters greater exploration, curbing susceptibility to local optima. Shared information from a common replay buffer and PSO algorithm substantially mitigates the computational load of training multiple agents. In the subsequent stage, TERL shifts to a refined exploitation phase. Here, only the best individual undergoes further refinement, while the remaining individuals continue PSO-based optimization. This allocates more computational resources to the best individual for yielding superior performance. Empirical assessments, conducted across a range of continuous control problems, validate the efficacy of the proposed TERL paradigm.

Introduction

Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm, finding applications in various domains (Berner et al. 2019; Vinyals et al. 2019; Moravčík et al. 2017) since the groundbreaking achievements of Deep Q-network (Mnih et al. 2015) and AlphaGo (Silver et al. 2017). While numerous reinforcement learning algorithms (RLs) (Schulman et al. 2015; Lillicrap et al. 2015; Van Hasselt, Guez, and Silver 2016; Fujimoto, Hoof, and Meger 2018) have been developed to address limitations in earlier approaches, such as value function approximation errors, several fundamental challenges persist. These challenges include sparse rewards and the issue of premature convergence to local optima (Khadka and Tumer 2018).

To address these challenges, Evolutionary Algorithms (EAs) have emerged as a promising alternative to traditional RL approaches (Salimans et al. 2017; Such et al. 2017). EAs evaluate an individual's fitness based on the cumulative rewards obtained in an episode, without being influenced by the frequency of rewards. However, due to this focus on the sum of rewards, EAs often suffer from lower sample efficiency compared to RL methods. Recognizing the potential synergies, Evolutionary Reinforcement Learning algorithms (ERLs) (Khadka and Tumer 2018; Pourchot and Sigaud 2019; Bodnar, Day, and Lió 2020; Hao et al. 2023) have been proposed, aiming to harness the strengths of both EAs and RL. ERL (Khadka and Tumer 2018), a popular approach in this domain, employs a dual strategy involving an actor population optimized using Genetic Algorithm (GA) and an additional actor optimized by an RL algorithm. This population of actors explores diverse regions of the solution space, enriching the training experiences for the RL actor. Furthermore, the updated RL actor is integrated back into the population, contributing to the next generation of individuals. This fusion of GA and RL within the ERL framework yields superior performance compared to standalone RL algorithms or GA. Some noteworthy variants of ERL, such as CEM-RL (Pourchot and Sigaud 2019), introduce distinct combination schemes like the Cross-Entropy Method (CEM) (Rubinstein and Kroese 2004) for population updates, while others, like Proximal Distilled ERL (PDERL) (Bodnar, Day, and Lió 2020), focus on more efficient genetic operators via stable distillation crossover and proximal mutation based on backpropagation. Additionally, ERL-Re^2 (Hao et al. 2023) adopts a two-scale representation to enhance the efficacy of GA and utilizes a value function approximator to boost sample efficiency. Numerous other enhanced methods and variations complement this vibrant landscape of research (Khadka et al. 2019; Majumdar et al. 2020; Suri 2022).

The majority of the aforementioned ERL approaches primarily employ actor networks as individuals of EA. However, this practice imposes limitations on the exploration capabilities of these individuals. For RL algorithms following actor-critic architecture, the actor network updates often rely on the value network. In many scenarios, the guidance for the update of the actor population comes from the RL actor. Consequently, the exploration potential of each

*Corresponding author.

individual is curtailed, heavily reliant on the sole RL critic network. To address this issue, we introduce the Two-stage Evolutionary Reinforcement Learning (TERL) method, designed to maintain a population consisting of both actor and critic networks. In the initial stage, every individual undergoes independent optimization through RL procedures, pursuing more exploration. Additionally, periodic updates to the population, executed through Particle Swarm Optimization (PSO), facilitate the sharing of information among individuals. While the cost of concurrently training multiple RL agents has hindered many ERL approaches from maintaining a population of complete RL agents, our approach overcomes this challenge through the information sharing of the common replay buffer and PSO updates. This first stage enables thorough exploration without excessive computational resources. In the subsequent stage, more computational resources are allocated to the best individual, aiming to achieve better final performance. Specifically, only the best individual is updated by RL algorithm, while the remaining individuals are updated by PSO, contributing diverse experiences to RL training. This staged approach emphasizes exploration efforts initially and shifts toward exploitation in the latter phase. The main contributions are as follows:

- **Novel Method TERL:** Our proposed TERL method maintains a population consisting of complete RL agents and divides the entire learning process into two distinct stages. In the earlier stage, all individuals engage in relatively independent and equitable exploration. In the latter stage, the best individual harnesses increased computational resources to attain enhanced performance.
- **Information Sharing Strategy:** We employ PSO alongside a shared replay buffer to facilitate the exchange of information among individuals. Demonstrating the power of this shared information, we showcase that individuals can reach the same performance as RL algorithms with significantly less interaction and update.
- **Implementation and Superior Performance:** We implement the TERL framework using the Twin delayed deep deterministic policy gradient algorithm (TD3) (Fujimoto, Hoof, and Meger 2018). Through comprehensive evaluation across various continuous control tasks, we demonstrate that TERL outperforms both state-of-the-art ERLs and RLs.

Background

To harness the synergies between EA and RL, many hybrid algorithms have emerged in the literature. These approaches can be broadly categorized into two types based on the nature of the individuals within a population: actor population and action population. Table 1 provides an overview of some representative ERLs, highlighting their distinctive characteristics.

A representative example of ERL is the one that maintains an actor population (Khadka and Tumer 2018). In this paradigm, a population of actors is optimized using GA, while a separate RL actor is refined using Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015). By integrating GA and DDPG, this configuration allows the diverse experiences generated by the actor population to augment the training of the RL actor. Additionally, the periodic inclusion of the actor back into the population provides new individuals. Although ERL has demonstrated the superiority of hybrid methods, its genetic operators are destructive and limit its performance. To address this concern, Proximal Distilled ERL (PDERL) (Bodnar, Day, and Lió 2020) introduces two genetic operators based on backpropagation: proximal mutation and distillation crossover. The former employs the sum of gradients to stabilize mutation strength, while the latter leverages parents' experiences to train offspring through Imitation Learning (Osa et al. 2018). These two operators are more stable than the original and improve ERL's performance. Another enhancement, Surrogate-assisted Controller (SC) (Wang et al. 2022), utilizes a surrogate model to estimate individual fitness, reducing sample inefficiency. Subsequently, ERL-Re^2 (Hao et al. 2023) presents a two-scale representation and utilizes GA to optimize the weights of the final linear layer of each individual, which can reduce GA's search space. Based on the proposed two-scale representation, it also employs a value function approximator to perform fitness estimation. To alleviate the sensitivity of parameters in RL algorithms, Collaborative ERL (CERL) (Khadka et al. 2019) employs policies with different hyperparameters to enable diverse exploration.

In addition to combining GA and DDPG-based RL algorithms, there are other combination schemes. CEM-RL (Pourchot and Sigaud 2019) proposes a different framework that integrates RL algorithms with the cross-entropy method (Rubinstein and Kroese 2004; Hansen 2016). CEM-RL applies RL gradient steps directly to half the population, without maintaining a separate RL actor. Compared to the combination mechanism in ERL, this scheme may be more suitable for CEM and achieves specific advantages over ERL. Evolution-based Soft Actor-Critic (ESAC) (Suri 2022) employs a similar framework to ERL, but it uses Evolution Strategies (ES) and SAC as its basic components. Additionally, ESAC proposes automatic mutation tuning to improve the algorithm's sensitivity to hyperparameters.

Another type of ERLs employs EA to search for superior actions to promote the learning of RL algorithms. Evolutionary Action Selection-TD3 (EAS-TD3) (Ma et al. 2022) utilizes PSO to evolve the action chosen by the policy network. Then, the evolved action is imitated by the policy network as target value. Cross-Entropy Guided Policies (CGP) (Simmons-Edler et al. 2019) employs CEM to search for optimal action to guide the learning of policy gradient methods. Additionally, a deterministic policy is trained by imitating the behavior of the CEM policy. Self-guided and self-regularized actor-critic (GRAC) (Shao et al. 2022) enhances RL learning by using an actor network to output the Gaussian distribution of actions and employing CEM to identify higher Q value actions.

Proposed Algorithm

The framework of TERL is illustrated in Fig. 1. This algorithm involves two stages, which focus more on explo-

Algorithm	individual type	RL method	how to update individuals	FP
ERL	actor	DDPG	actor insertion, n-point crossover, Gaussian mutation	×
PDERL	actor	DDPG	actor insertion, distillation crossover, proximal mutation	×
SERL/SPDERL	actor	DDPG	actor insertion, n-point crossover, Gaussian mutation	✓
ERL-Re^2	actor(partial)	TD3	actor insertion, n-point crossover, Gaussian mutation	✓
CERL	actor	TD3	actor insertion, n-point crossover, Gaussian mutation	×
CEM-RL	actor	TD3	RL gradient, CEM	×
ESAC	actor	SAC	actor insertion, hindsight crossover, ES	×
EAS-TD3	action	TD3	PSO	×
CGP	action	TD3	CEM	×
GRAC	action	PG	CEM	×

Table 1: A comparison of some representative ERLs. FP stands for Fitness Prediction.

Algorithm 1: TERL

Input: population size n;
maximum timesteps m;
exploration ratio $ratio$;
Output: best individual P_b

Initialize a population P with random value;
Initialize learned_steps L, best_fitness B, timestep t, index e and b with zero;
while $t < m$ **do**
  if $t < m \times ratio$ **then**
    $stage = 1$;
  else
    $stage = 2$;
  $index\_list = \{\overbrace{e, e, ..., e}^{n}, 1, 2, ..., n\}$;
  for i *in* $index\_list$ **do**
    $fitness, step$ = Evaluate($actor$ of P_i, R);
    $t = t + step$;
    $L_i = L_i + step$;
    if $fitness > B_i$ **then**
      $B_i = fitness$;
      $L_i = 0$;
      $e = i$;
    if $B_i > max(B)$ **then**
      if *stage is 1* **then**
        $b = i$;
      else
        Copy the actor of P_i to P_b;
    if *stage is 1* **then**
      Optimize individual P_i by RL method;
    else
      Optimize individual P_b by RL method;
  if $L_e > L_b$ *or stage is 2* **then**
    $e = b$;
  Update actors in population by PSO;
return P_b;

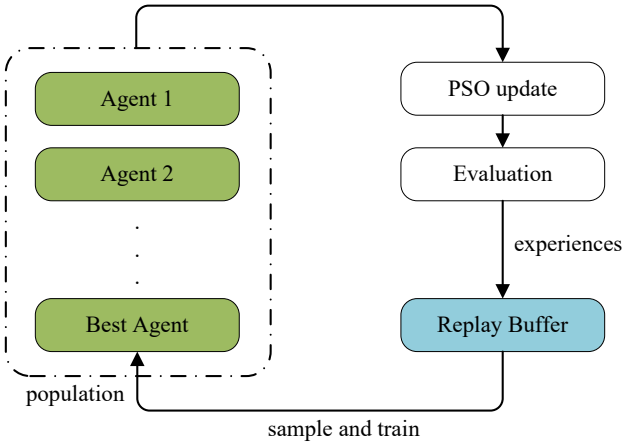


Figure 1: The general framework of TERL. In the exploration stage, each individual is trained respectively; in the exploitation stage, only the best individual is trained.

ration and exploitation respectively. Each iteration of the two stages consists of several main steps, including population evaluation, PSO-driven population updates, and gradient-based individual updates. One main difference between the two stages lies in the learning focus: the early stage emphasizes comprehensive training of all individuals, while the latter stage focuses on training the best individual. Apart from that, there are other differences that are detailed in the following.

Exploration Stage

In this stage, independent exploration is promoted among all individuals. Each individual maintains and trains their own actor and critic networks, which allows actors to explore different domains of solution space without the limitation of the same critic. To reduce the computation burden of training multiple agents concurrently and accelerate the learning process, individuals share their learned information through the common replay buffer and periodic PSO updates. The information flow in the exploration stage is depicted in Fig. 2. As shown in Fig. 2, the experiences of all individuals' ac-

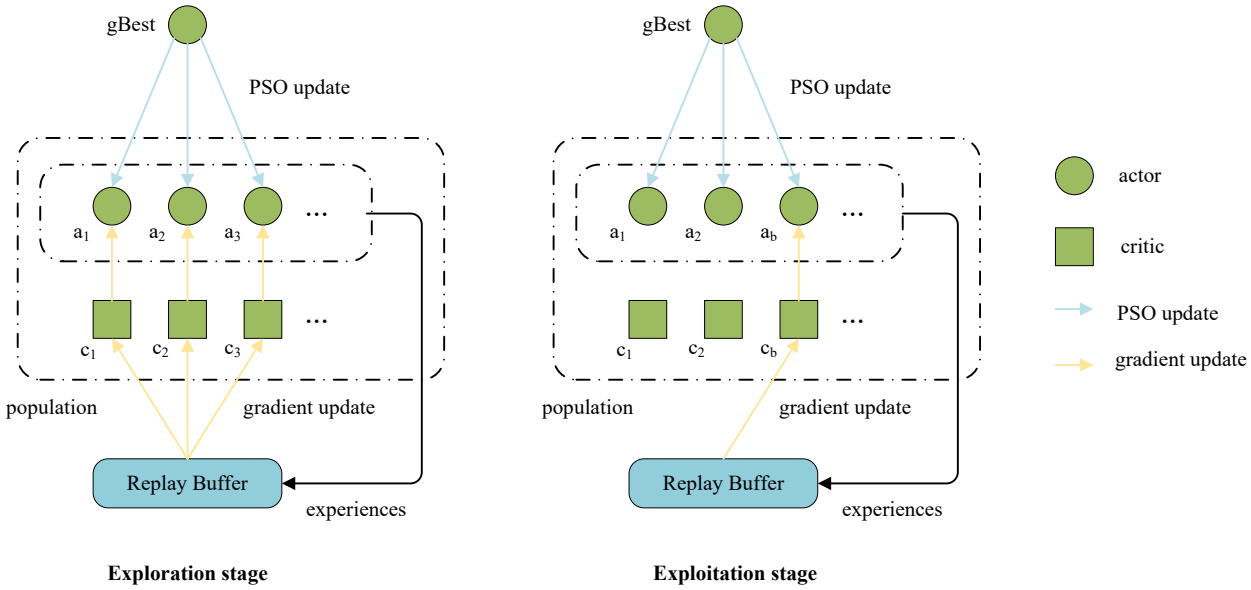


Figure 2: The information flow in TERL.

Algorithm 2: Function Evaluation

Input: actor π;
replay buffer R;
Output: $fitness, step$

1 Initialize $fitness = 0$, $step = 0$;
2 Reset environment and get initial state s_0;
3 **while** *env is not done* **do**
4 Select action $a_t = \pi(s_t|\theta^\pi) + \epsilon, \epsilon \sim N(0, \sigma)$;
5 Execute action a_t, get reward r_t and new state s_{t+1};
6 $fitness = fitness + r_t$;
7 Append transition (s_t, a_t, r_t, s_{t+1}) to R;
8 $step = step + 1$
9 **return** $fitness$, $step$;

tors are stored in the replay buffer and then experiences are sampled to train the critic of every individual respectively. After that, every individual's actor is updated based on the guidance from its critic. Moreover, every individual's actor can learn the information from the global best individual by PSO update. It is worth noting that we restrict the update frequency of PSO for the purpose of preserving the independence of individuals. In this manner, we make a trade-off between individuals' independence and information sharing.

The pseudocode of our method is provided in Algorithm 1. When the number of interaction steps is less than a certain ratio of the total, we consider this phase as the initial exploration stage (lines 4-5). This stage mainly includes three procedures: evaluation, gradient update and PSO update.

After the evaluation of an individual, the corresponding gradient update is executed with the same number of interaction steps. Unlike previous ERLs, we do not evaluate every individual the same number of times. As shown in line 8 in Algorithm 1, an additional n evaluations are allocated to the individual with index e prior to the population evaluation. In this stage, e indicates the individual that makes improvement most recently. We evaluate and optimize these individuals rather than the current best-performing individual more times to encourage exploration. Arrays B and L are used to record the historically best fitness and the elapsed timesteps after reaching it, which are updated after evaluation (lines 12-15). Index b indicates the current best-performing individual. If individual e accumulates more timesteps than the current best b, the individual e is superseded by b (lines 26-27).

Following the evaluation and gradient update procedures, the PSO update is executed at the end of an iteration (line 28). Considering that individuals are jointly optimized by RL and PSO algorithms, to avoid conflicts between both, the current individual is regarded as its historically best individual. In addition, the PSO update is only applied to the actor network and executed periodically (usually 10,000 timesteps in this stage) to maintain the individuals' dependence.

Exploitation Stage

In the exploitation stage, the best individual leverages additional computational resources, while other individuals no longer maintain and optimize their critic network. This choice arises from the understanding that, despite the benefits of information sharing between individuals in lightening computational burdens for multiagent training, the total resources used are still more than training one agent. In addition, compared to the earlier phase, agents learn slowly and require more interaction and optimization in the latter phase. Therefore, to achieve a better final performance, only the best individual is trained by RL while other individuals are optimized by PSO to provide diverse experiences.

The information flow in the exploitation stage is depicted in Fig. 2. After collecting the experiences of all individuals' actors, only the actor and critic of the best individual are updated with gradients that are calculated by the sampled experiences. Then, similar to the prior phase, actors of all individuals are optimized with the information from the $gBest$.

As shown in Algorithm 1, when the number of interaction steps is larger than a certain ratio of the total, this phase is regarded as the latter exploitation stage (lines 6-7). The main steps in this period are similar to those in the exploration stage: evaluation, gradient update and PSO update.

Distinct from the prior evaluation procedure, the index e always denotes the best individual P_b (lines 26-27) in this phase, which means that P_b is always evaluated extra. In this way, the best individual can have more opportunities to interact with environment and explore unknown areas. After the evaluation of P_i, if P_i's fitness is greater than the previous best fitness, the parameters of P_i's actor will be copied to P_b's actor (lines 20-21). This operation can share the learned information of other individuals with P_b. Additionally, this information is propagated to the entire population through the subsequent PSO update. Unlike the last stage, the index b remains unchanged in this stage, that is, the "best" individual will not be replaced after the exploration stage. Even if other individuals achieve a higher reward, their value networks no longer adapt to their actors due to a long period without updating and cannot guide the updating of actors in subsequent training. For the following gradient update, we no longer train all individuals but instead allocate most resources to train the best individual. Therefore, after the evaluation of any individual, only the best individual is optimized by RL method (lines 24-25).

Finally, the population is updated by PSO, which is more frequent in this stage (usually every 1,000 timesteps). The main purpose is to promote information exchange among individuals when other individuals are not updated by gradient methods.

Experimental Studies

Test Problems

To evaluate the effectiveness of proposed TERL, we utilize the popular Mujoco simulation environments as test problems (Todorov, Erez, and Tassa 2012), which are widely used in RL fields (Schulman et al. 2017; Lillicrap et al. 2015; Duan et al. 2016). These benchmarks are available through OpenAI gym (Brockman et al. 2016), which provides an interface for researchers to benchmark their algorithms. The goal of these problems is to apply torque on the robots' joints to complete specific locomotion tasks. Apart from evaluating our method on Mujoco environments, we also apply it to three classic control problems to test its performance more fully.

Performance Metric

In this study, the performance of agents is measured by the sum of rewards in one episode. For population-based algorithms, the individual with the highest fitness within the population is selected and tested. To evaluate RL algorithms, the performance of actor is tested every five thousand steps. To ensure a fair comparison between RL algorithms and population-based algorithms, the steps of every individual in the population are accumulated. Additionally, each candidate is tested for five episodes, and the average score is recorded as the algorithm's performance. Finally, to ensure statistical robustness, the reported results are the average scores from five independent runs with different random seeds.

Experimental Setting

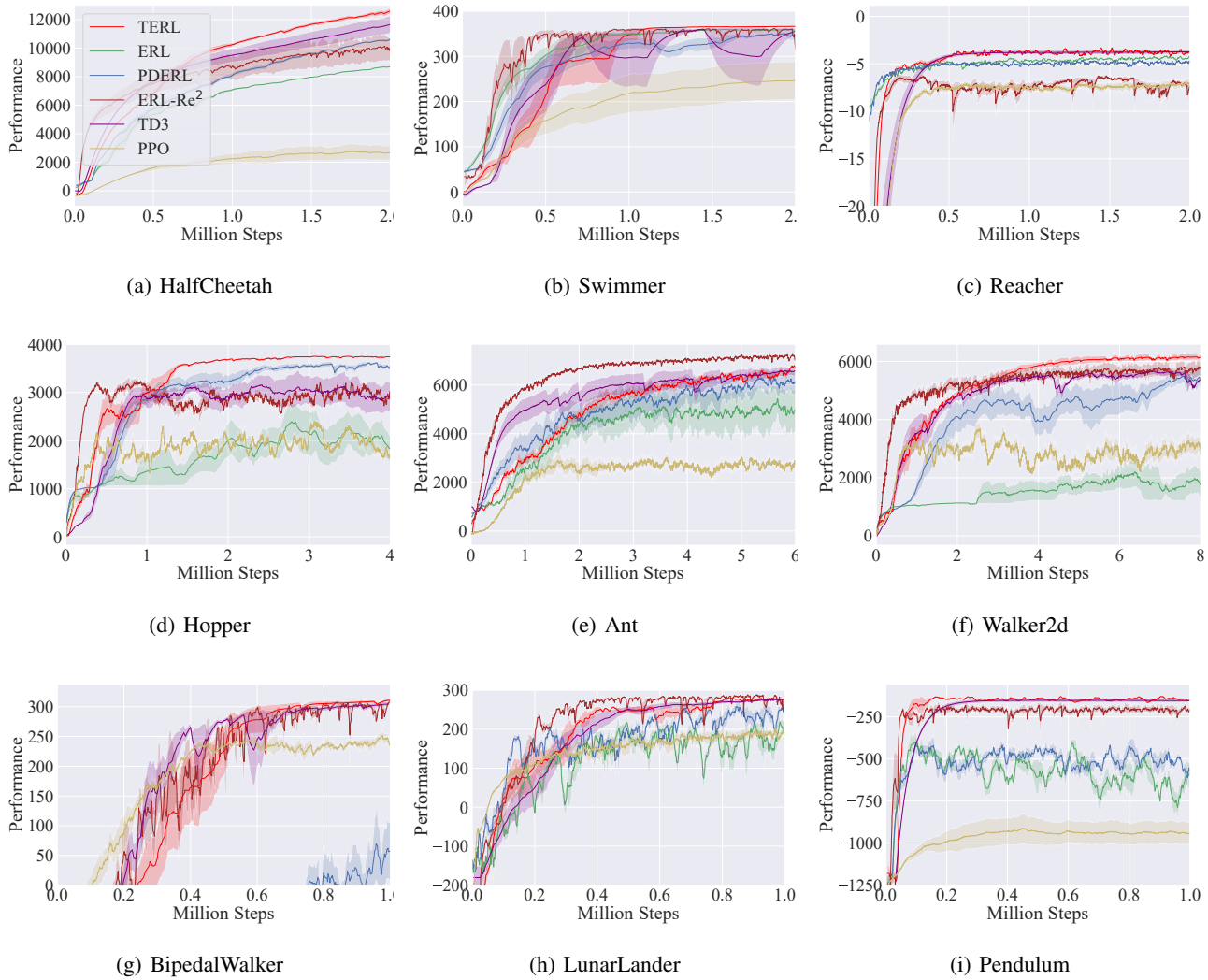
To assess the practical efficacy of the proposed algorithm, we conducted a comparative analysis against state-of-the-art ERLs (ERL (Khadka and Tumer 2018), PDERL (Bodnar, Day, and Lió 2020), ERL-Re^2 (Hao et al. 2023)), as well as RL algorithms (TD3 (Fujimoto, Hoof, and Meger 2018) and PPO (Schulman et al. 2017)). All compared algorithms, including ERL, PDERL, ERL-Re^2, TD3 and PPO, are run using their published implementations. We set the hyperparameters of each algorithm to the values suggested in the corresponding papers or default values in the authors' implementation.

The unique hyperparameters of our method used in experiments are detailed as follows. The exploration ratio is 0.25 in all environments. The inertia weight w is 0 and both acceleration coefficients c_1 and c_2 are 1 in PSO. The update frequency of PSO is 10,000 timesteps in the earlier stage and 1,000 timesteps in the latter stage.

Experimental Results

Fig. 3 depicts the learning curves of the compared algorithms in solving nine continuous control problems, with the shaded areas indicating standard variation of performance. In addition, Table 2 presents the mean and standard deviation of the final performances. Overall, TERL outperforms the other methods in most environments except Ant and LunarLander. In these two environments, the performance of our method is slightly inferior to ERL-Re^2 and still the second best in all compared algorithms.

As shown in Fig. 3, the reported performance of TERL is not very outstanding during the earlier exploration period, mainly due to the training of multiple agents. However, our method demonstrates faster learning than other algorithms in the latter period after more exploration. Compared to TD3, our method outperforms it in all environments, which can be attributed to the exploration ability of population. By the collaborative search and information exchange of multiple individuals, our method is less likely to fall into local optima, leading to better final performance. In contrast to ERL-Re^2, our performance is inferior on Ant and LunarLander, which may be due to the high sample efficiency of its value function approximator. Nevertheless, this approximator requires additional training and significantly slows down the training speed of algorithm. Therefore, our method is noticeably faster than ERL-Re^2, which is discussed in the supplementary materials.



(a) HalfCheetah (b) Swimmer (c) Reacher

(d) Hopper (e) Ant (f) Walker2d

(g) BipedalWalker (h) LunarLander (i) Pendulum

Figure 3: Learning curves on Mujoco continuous control benchmarks and classic control problems

The Effects of Two Stages

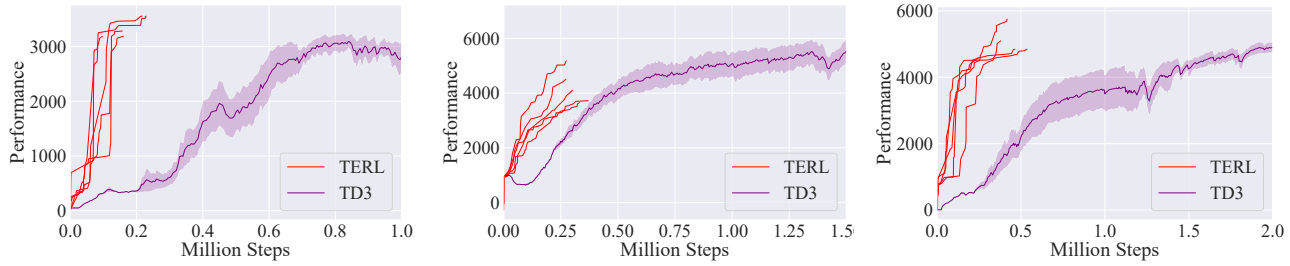
In this section, first, we demonstrate the effect of information sharing by individuals' learning curves in the exploration stage. Then, ablation experiments are conducted to validate the effectiveness of the two stages. All experiments in this section are carried out in three relatively more difficult environments: Hopper, Ant, and Walker.

Fig. 4 depicts the learning curves of the best individual during the exploration stage. Different curves of our method are derived from five independent runs. Because the exploration ratio is 0.25 in the experiments, the number of timesteps for the compared TD3 is also 25% of the maximum number of timesteps in Fig. 4. As shown in Fig. 4, the best individual can achieve a performance that approaches or even exceeds that of a single TD3 agent with only approximately 20% interaction and training. The only difference between individuals of our method and a single TD3 agent is that our individuals employ the shared information from the common replay buffer and PSO update. Therefore, information sharing among individuals can significantly reduce the resource usage of training multiple agents in earlier periods and alleviate the concern that extra interaction and gradient updates for multiple agents may decrease the final performance of the algorithm.

Furthermore, we conduct ablation experiments with variants that only have the exploration or exploitation stage to enable a comparison with the original TERL. We record the performance of both variants in five independent runs with different random seeds and plot the results in Figure 5. As shown in Figure 5, the original TERL achieves better final performance than both variants. For the variant without the exploration stage, although it learns significantly faster than TERL in the earlier period, its performance is surpassed in later stages. In addition, the performance gap between the variant without exploitation and the original algorithm is not very significant, which is mainly because the extra evalua-

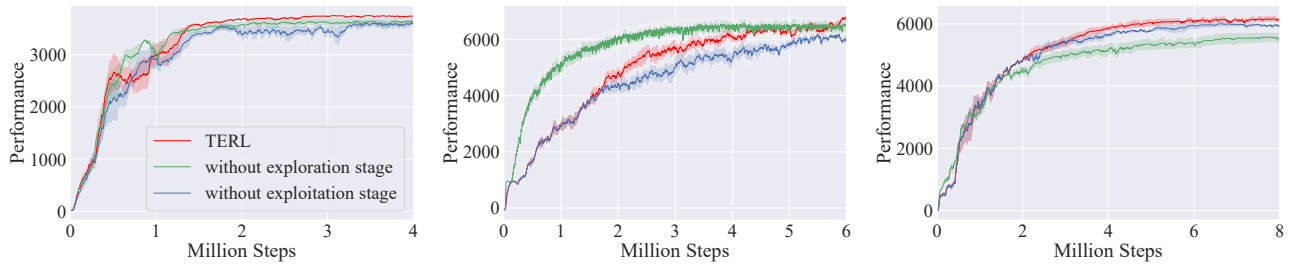
	Environment	TERL	ERL	PDERL	ERL-Re^2	TD3	PPO
Mujoco	HalfCheetah	**12632 ± 400**	8697 ± 108	10622 ± 221	9991 ± 2089	11880 ± 1327	2660 ± 1251
	Swimmer	**366 ± 2**	358 ± 4	347 ± 13	314 ± 88	354 ± 19	246 ± 99
	Reacher	**-3.7 ± 0.4**	−4.3 ± 0.3	−4.6 ± 0.3	−7.3 ± 1.3	**-3.7 ± 0.5**	−7.0 ± 1.1
	Hopper	**3736 ± 24**	1850 ± 608	3546 ± 97	3044 ± 130	2900 ± 650	1656 ± 278
	Ant	6611 ± 291	4996 ± 1991	6116 ± 1075	**7210 ± 93**	6552 ± 419	2872 ± 295
	Walker2d	**6149 ± 219**	1666 ± 835	5425 ± 696	5807 ± 582	5386 ± 227	2822 ± 646
Classical	BipedalWalker	**311 ± 8**	−25 ± 31	54 ± 107	304 ± 11	304 ± 9	237 ± 15
	LunarLander	277 ± 5	194 ± 34	248 ± 17	**283 ± 4**	273 ± 7	188 ± 24
	Pendulum	**-148 ± 18**	−581 ± 178	−564 ± 103	−201 ± 40	−153 ± 24	−937 ± 156

Table 2: The mean and standard variation of final performance in Mujoco test environments and classic control benchmarks



(a) Hopper (b) Ant (c) Walker2d

Figure 4: Comparison between the learning curves of the best individual in the exploration stage and single TD3 agent. Different curves with the same color represent the results of different independent experiments.



(a) Hopper (b) Ant (c) Walker2d

Figure 5: Ablation experiments with the variants that only have the exploration or exploitation stage.

tion and update for the best individual in our algorithm can achieve a similar effect to the exploitation stage.

Conclusions

In this paper, we present TERL, a two-stage ERL that emphasizes exploration and exploitation. In the initial exploration stage, all individuals in the population maintain and train their own actor and critic networks. To reduce the computational burden of training multiple agents simultaneously, a common replay buffer and PSO update are used to share the information among individuals. Subsequently, the population enters the exploitation stage, where the best individual is trained using RL methods. Other individuals undergo more frequent PSO updates, enriching the best individual's experiences. TERL is realized with TD3 and is evaluated against state-of-the-art algorithms in continuous control tasks, demonstrating its effectiveness.

Regarding future work, exploring alternative selection methods, such as Novelty Search (Lehman and Stanley 2011; Conti et al. 2018) or Quality Diversity methods (Pugh, Soros, and Stanley 2016; Mouret and Clune 2015; Cully and Demiris 2017), has potential to enrich experiences for RL.

Acknowledgments

This work was supported by the National Natural Science Funds for Distinguished Young Scholar under Grant 62325307, in part by the National Natural Science Foundation of China under Grants 62203308, 62173236, 62073225 and 62203134, in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076, and in part by the Shenzhen Science and Technology Program under Grants JCYJ20220531101411027 and 20220809141216003, and in part by the Scientific Instrument Developing Project of Shenzhen University under Grant 2023YQ019.

References

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

Bodnar, C.; Day, B.; and Lió, P. 2020. Proximal distilled evolutionary reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3283–3290.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Conti, E.; Madhavan, V.; Petroski Such, F.; Lehman, J.; Stanley, K.; and Clune, J. 2018. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in neural information processing systems*, 31.

Cully, A.; and Demiris, Y. 2017. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2): 245–259.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, 1329–1338. PMLR.

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

Hansen, N. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Hao, J.; Li, P.; Tang, H.; Zheng, Y.; Fu, X.; and Meng, Z. 2023. ERL-Re^2: Efficient Evolutionary Reinforcement Learning with Shared State Representation and Individual Policy Representation. *International Conference on Learning Representations*.

Khadka, S.; Majumdar, S.; Nassar, T.; Dwiel, Z.; Tumer, E.; Miret, S.; Liu, Y.; and Tumer, K. 2019. Collaborative evolutionary reinforcement learning. In *International conference on machine learning*, 3341–3350. PMLR.

Khadka, S.; and Tumer, K. 2018. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, 31.

Lehman, J.; and Stanley, K. O. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2): 189–223.

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

Ma, Y.; Liu, T.; Wei, B.; Liu, Y.; Xu, K.; and Li, W. 2022. Evolutionary Action Selection for Gradient-Based Policy Learning. In *International Conference on Neural Information Processing*, 579–590. Springer.

Majumdar, S.; Khadka, S.; Miret, S.; McAleer, S.; and Tumer, K. 2020. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *International Conference on Machine Learning*, 6651–6660. PMLR.

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

Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337): 508–513.

Mouret, J.-B.; and Clune, J. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Osa, T.; Pajarinen, J.; Neumann, G.; Bagnell, J. A.; Abbeel, P.; Peters, J.; et al. 2018. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179.

Pourchot, A.; and Sigaud, O. 2019. CEM-RL: Combining evolutionary and gradient-based methods for policy search. *International Conference on Learning Representations*.

Pugh, J. K.; Soros, L. B.; and Stanley, K. O. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 40.

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

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

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

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

Shao, L.; You, Y.; Yan, M.; Yuan, S.; Sun, Q.; and Bohg, J. 2022. Grac: Self-guided and self-regularized actor-critic. In *Conference on Robot Learning*, 267–276. PMLR.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Simmons-Edler, R.; Eisner, B.; Mitchell, E.; Seung, S.; and Lee, D. 2019. Q-learning for continuous actions with cross-entropy guided policies. *arXiv preprint arXiv:1903.10605*.

Such, F. P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K. O.; and Clune, J. 2017. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.

Suri, K. 2022. Off-Policy Evolutionary Reinforcement Learning with Maximum Mutations. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 1237–1245.

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

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, Y.; Zhang, T.; Chang, Y.; Wang, X.; Liang, B.; and Yuan, B. 2022. A surrogate-assisted controller for expensive evolutionary reinforcement learning. *Information Sciences*, 616: 539–557.