2025 lEEE 14th Data Driven Control and Learning Systems Conference
May 9-11, 2025, Wuxi, China

A proximal policy optimization algorithm based on adaptive hierarchical particle swarm

Jianfei Wei, Guoqing Xie, Longlong Wang, Guangyu Wang, Shuping He, Chengcheng Ren, Liang Zhang*
Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, School of Electrical Engineering and Automation, Anhui 230601, P. R. China
E-mail: liangzhang@ahu.edu.cn

Abstract: This paper proposes an adaptive hierarchical particle swarm optimization enhanced proximal policy optimization (AHPSO-PPO) framework. This framework addresses key challenges in reinforcement learning (RL), particularly slow convergence and premature convergence to sub-optimal policies. Firstly, adaptive parameter tuning, hierarchical particle swarm optimization, and variational evolution strategies are introduced in AHPSO. Secondly, the network parameters in RL are used as the optimization objective of the intelligent algorithm to obtain the optimal parameters faster and better. Finally, the experimental verification in a simulation environment shows that the AHPSO-PPO algorithm proposed in this paper performs better than standard PPO and DDPG. It has a faster convergence speed and a certain degree of robustness to environmental randomness. This hybrid approach links gradient-based optimization with group intelligence, providing a scalable solution for real-world decision-making systems.

Key Words: Deep Reinforcement Learning, Particle swarm optimization algorithm, Hybrid Optimization Algorithm, Artificial intelligence

2025 IEEE 14th Data Driven Control and Learning Systems (DDCLS) | 979-8-3503-5731-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/DDCLS66240.2025.11065850

1 Introduction

Reinforcement learning(RL)[1] represents a cornerstone paradigm within machine learning frameworks[2], tailored explicitly for sequential decision-making under uncertainty. The problem formulation typically employs Markov Decision Processes (MDPs)[3] as its mathematical foundation, enabling RL agents to systematically refine decision policies through alternating cycles of value function evaluation and gradient-based optimization[4]. The integration of deep neural networks has revolutionized RL capabilities[5], allowing approximation of high-dimensional value functions while demonstrating superior sample efficiency compared to traditional gradient-free methods. The development of off-policy RL algorithms further enhances sample efficiency through strategic reuse of experiential data[6], making them particularly advantageous in scenarios with costly data acquisition. These methodological advances have catalyzed breakthrough achievements across diverse domains, including strategic game mastery[7], robotic motion control[8], adaptive recommendation systems[9], and combinatorial scheduling[10]. Despite these successes, RL systems still confront significant challenges, such as inefficient exploration-exploitation tradeoffs in high-dimensional state spaces[11], suboptimal convergence characteristics during policy optimization[12], pronounced sensitivity to hyperparameter selection[13], and inherent limitations in gradient-based optimization dynamics[14]. These unresolved issues collectively impede RL's deployment in safety-critical real-world applications requiring robust performance guarantees.

Two prominent RL algorithms that have garnered significant attention in recent years are Asynchronous Advantage Actor-Critic (A3C)[15] and Proximal Policy Optimization (PPO)[16]. A3C, an actor-critic algorithm developed by DeepMind in 2016, improves the stability and performance of traditional RL methods by employing asynchronous gradient descent to optimize deep neural network controllers. Its distributed architecture parallelizes training, accelerating convergence through simultaneous environment exploration. In contrast, PPO, a policy gradient method introduced by OpenAI in 2017, simplifies agent training while enhancing reliability. By balancing exploration-exploitation trade-offs and enforcing stable policy updates, PPO utilizes a clipping mechanism to constrain adjustments within a trust region. This prevents destabilizing policy deviations, ensuring consistent performance during optimization.

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart[17] in 1995, which drew inspiration from avian collective movement patterns. This evolutionary computation technique simulates particle swarm dynamics to locate optimal solutions through distributed collective intelligence[18]. Over three decades of development, PSO has demonstrated versatility across computational domains, with proven efficacy in function optimization[19], machine learning parameter search[20], robotic motion planning[21], and neural architecture optimization[22].

The integration of PSO with RL seeks to augment RL's optimization efficacy in complex, high-dimensional problem domains. Conventional RL approaches frequently encounter inherent limitations such as suboptimal convergence rates and local optima entrapment during policy training. As a global optimization metaheuristic, PSO addresses these challenges through swarm intelligence principles, employing population-based exploration strategies that systematically navigate solution spaces while maintaining global convergence properties. This synergy particularly enhances RL's capacity to efficiently traverse high-dimensional state-action spaces, thereby improving decision-making precision in complex tasks. The combined framework has shown particular promise in dynamic applications requiring real-time

This work was supported in part by the State Key Laboratory of Micro-Spacecraft Rapid Design and Intelligent Cluster under grant No. MS01240110, the Anhui Provincial Key Research and Development Project under Grant 2022i01020013, the University Synergy Innovation Program of Anhui Province under Grant GXXT-2021-010, the Anhui Provincial Natural Science Foundation Youth Project under Grant 2408085QF204, and the Anhui Higher Education Scientific Research Project under Grants 2024AH050061, 2022AH050068.

adaptation, including robotic motion planning, autonomous navigation systems, and constrained control optimization.

Recent research highlights the complementary strengths of PSO-RL integration across multiple domains. Wang et al.[23] reformulated RL problems as metaheuristic optimization tasks, utilizing PSO's global search capabilities to identify Pareto-optimal solutions. In energy-constrained systems, Choudhary et al.[24] developed a PSO-RL hybrid framework that improves wireless sensor network reliability through optimized cluster head selection and fault detection mechanisms. Ding et al.[25] asynchronous PSO variant synergized with backward Q-learning, establishing a novel asynchronous RL framework based on the Sarsa algorithm. For continuous control challenges, Liu et al.[26] proposed PG-PSOPE, a gradient-free policy optimization method that employs PSO for parameter space exploration, achieving variance reduction and accelerated policy network convergence. Expanding this paradigm, Zhang et al.[27] implemented a multi-swarm framework combining PSO with M3DDPG, where coordinated particle swarms generate diversified training samples, demonstrating enhanced training efficiency with accelerated convergence rates relative to baseline methods.

This study introduces a novel hybrid architecture combining adaptive hierarchical particle swarm optimization enhanced proximal policy optimization(AHPSO-PPO). The proposed framework synergistically embeds hierarchical swarm intelligence mechanisms into PPO's policy update paradigm, utilizing our enhanced AHPSO variant to dynamically optimize neural policy parameters during gradient-based learning iterations. This integration specifically mitigates inherent limitations of conventional gradient optimization in RL, including local optima entrapment and convergence instability, while achieving accelerated training dynamics with enhanced precision in complex environments.

2 Basic Theory

2.1 Reinforcement Learning

We formalize the interaction between an agent and an environment ε as a sequential decision-making process over discrete time steps. At each step, the agent observes the current state s and selects a valid action a according to its deterministic policy $\pi : S \rightarrow A$, which maps states to actions. After executing a, the agent transitions to a new state $s^{'}$ governed by the environment's dynamics, and receives a scalar reward signal r from ε. This reward informs the agent's iterative policy refinement process, where it adjusts its action-selection strategy to maximize cumulative rewards over time.

To exemplify the reinforcement learning (RL) framework, we adopt the classic video game *Sokoban* as a case study. Within this context, the environment ε corresponds to the game engine, where the state s is defined by the current screen display. During gameplay, the agent iteratively selects actions (e.g., moving left, right, down, or up) from the action space A to navigate the environment and push crates toward predefined targets, thereby maximizing cumulative rewards. The reward signal r is quantified as the player's score, which serves as feedback for the agent's action-selection policy $\pi(s, a)$. This policy determines the probability or preference of choosing action a in state s to optimize long-term objectives. The interplay between the agent, environment, and policy is summarized in the RL architecture depicted in Fig. 1.

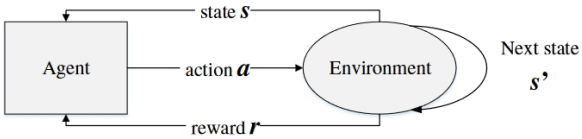


Fig. 1: The Framework of Reinforcement Learning

2.2 Proximal Policy Optimization Algorithm

PPO is a policy-based reinforcement learning algorithm designed to enhance the Policy Gradient (PG) framework while preserving its efficacy in continuous state and action spaces. As an on-policy algorithm, PG relies on the current policy π_θ for both trajectory generation and policy updates, ensuring alignment between sampled data and the learning objective. However, PG suffers from high sensitivity to step size selection, which often leads to unstable gradient updates and poses significant challenges for parameter optimization. The policy gradient is defined in formula (1):

$$\nabla \overline{R}_\theta = \hat{E}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) R_t \right] \tag{1}$$

where π_θ denotes the stochastic policy distribution, and R_t represents the cumulative reward.

To overcome the constraints of policy gradient methods, PPO incorporates two methodological innovations: (1) an experience replay mechanism that facilitates multi-update data utilization through off-policy correction, and (2) a clipped probability ratio constraint that mitigates policy divergence during optimization updates.

The off-policy paradigm operates under the condition that the behavioral policy and target policy maintain distinct parameters. The importance sampling technique provides theoretical guarantees for cross-policy updates, enabling parameter transfer from the sampling network θ to the training network $\theta^{'}$ through variance-reduced gradient estimation.

Suppose there is a function $f(x)$, we use data sampled from distribution q to estimate the expected value of function $f(x)$ with distribution p. Since the data are sampled from q, we use the equation in formula (2) to calculate the expected value.

$$\begin{aligned} E_{x\sim p}\left[f(x)\right] &= \int f(x)\frac{p(x)}{q(x)}q(x)dx \\ &= E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] \end{aligned} \tag{2}$$

where $\frac{p(x)}{q(x)}$ is called the importance weight and can be used to correct the distribution.

So, the definition of policy gradient is shown in formula (3).

$$\nabla \overline{R}_\theta = \hat{E}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta'}(a_t \mid s_t)} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A^{\theta'}(s_t, a_t) \right] \tag{3}$$

where (s_t, a_t) is sampled by the policy $\pi_{\theta'}$. $A^{\theta'}(s_t, a_t)$ is the advantage function.

To stabilize policy updates through bounded policy divergence, a clipped surrogate function is introduced as per Equation 4. The truncation mechanism actively limits the maximum allowable deviation between current and updated policies during gradient-based optimization, serving as a safeguard against unstable learning dynamics.

$$J_{\text{PPO}}^{\text{CLIP}}(\theta) = \sum_{(s_t,a_t)} \min \left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta'}(a_t \mid s_t)} A^{\theta'}(s_t, a_t), \right. \\ \left. \text{clip}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta'}(a_t \mid s_t)}, 1-\varepsilon, 1+\varepsilon \right) A^{\theta'}(s_t, a_t) \right) \tag{4}$$

where ε is a parameter, and in this paper, the value is 0.2.

The restricted range diagram of the objective function is shown in Fig. 2, and the abscissa in the figure is $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$.

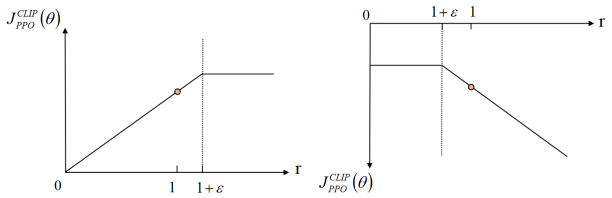


Fig. 2: Restricted range of objective function

The parameter-shared architecture integrating policy and value networks augments its composite objective function with two critical components: (1) value estimation error minimization for precise credit assignment, and (2) policy entropy regularization that maintains stochastic policy diversity, jointly optimizing exploration-exploitation dynamics while ensuring learning stability. The objective function is shown in formula (5).

$$J_{\text{PPO}}(\theta) = E_t \Big[J_{\text{PPO}}^{\text{CLIP}}(\theta) - c_1 (V_\theta(s) - V_{\text{target}})^2 \\ + c_2 H(s, \pi_\theta) \Big] \tag{5}$$

where c_1, c_2, are parameters. $(V_\theta(s) - V_{\text{target}})^2$ is the mean square error of the value function. $H(s, \pi_\theta)$is the entropy value of policy π_θ.

2.3 Particle swarm optimization algorithm

PSO, as a typical swarm intelligence method, is inspired by the bionic principle of the flock foraging behavior of birds. The basic process of the algorithm starts by randomly initializing a set of potential solutions (called individual particles), followed by a dual bootstrapping mechanism to dynamically correct the particle states during the iteration process. Each particle, based on the synergistic effect of its historical optimal solution P_{best} and the global optimal solution of the group G_{best}, can explore the space by means of the equations (6) and (7) defined by the velocity and position update equations to achieve spatial exploration. This unique collaborative model organically combines individual cognition and group intelligence to drive the particle swarm to gradually approach the optimal region in the solution space.

$$v_i \leftarrow w v_i + c_1 r_1 (p_i - x_i) + c_2 r_2 (p_g - x_i) \tag{6}$$

$$x_i \leftarrow x_i + v_i \tag{7}$$

where $\nu_i \in [\nu_{\min}, \nu_{\max}]$ and $x_i \in [x_{\min}, x_{\max}]$ denote the velocity and position of the ith particle, w denotes the inertia factor, c_1 and c_2 are two hyper parameters of the algorithm, p_i denotes the individual optimal value of the ith particle, p_g denotes the global optimal value of global, and r_1, r_2 are two random number between 0 and 1.

3 Adaptive Hierarchical Particle Swarm Optimization Algorithm

To enhance the global exploration performance and solution accuracy of PSO, this study introduces an enhanced approach incorporating self-adaptive parameter tuning, hierarchical particle optimization, and combined mutation operators. This algorithm establishes a balance between exploration and development through adaptive coefficient modulation within the framework, and collaborates with particle swarm intelligence to optimize dynamic search capabilities, while combining composite mutation operators to address population stagnation in evolutionary computation.

3.1 Adaptive parameter adjustment strategy

The algorithm adopts dynamically adjusted inertia weights and learning factors to achieve balanced control between exploration and development. Assuming the current iteration count is t and the maximum iteration count is T, the main parameters will change according to the following pattern:

$$w(t) = w_{\max} - (w_{\max} - w_{\min}) \frac{t}{T} \tag{8}$$

$$c_1(t) = c_1^{\max} - (c_1^{\max} - c_1^{\min}) \frac{t}{T} \tag{9}$$

$$c_2(t) = c_2^{\min} + (c_2^{\max} - c_2^{\min}) \frac{t}{T} \tag{10}$$

where w is the inertia influence factor, $w_{\max}$ and $w_{\min}$ are the upper and lower bounds of the inertia influence factor, respectively, which mainly affect the speed and pace of the particle movement. When the inertia influence factor is larger, the speed influence of the particle is also larger, and the exploration range is wider. When the inertia influence factor is smaller, the speed influence of the particle is reduced, and the search is finer, and w will decrease linearly with the increase of the iteration number. c_1 is the individual learning factor, $c_1^{\max}$ and $c_1^{\min}$ are the upper and lower bounds of the individual learning factor, c_1 decreases with the increase of iterations, which is in line with the requirement that the algorithm pays attention to the individual search in the early stage. c_2 is the social learning factor, $c_2^{\max}$ and $c_2^{\min}$ are the upper and lower bounds of the social learning factor, and c_2 increases with the number of iterations, which meets the requirement of the algorithm to pay more attention to the collective learning experience in the later stage.

3.2 Particle swarm hierarchical optimization strategy

In each algorithm cycle, particles receive unique fitness scores indicating their search space quality. The swarm is then sorted and divided into high-quality (20%), common (60%), and low-quality (20%) groups, each updated using specialized formulas.

During the continuous search of particles, the high-quality particle swarm is closer to the optimal region in the solution space, while the low-quality particle swarm may be more dispersed compared to the high-quality particle swarm, and thus the low-quality particle swarm should concentrate on exploring the space.

In this paper, we propose a strategy of using a high-quality particle swarm instead of a single optimal particle to guide the update of inferior particles. Assuming that there exists a particle swarm with n particles, in which the number of particles in the high-quality particle swarm (GP) is α, the number of particles in the common particle swarm (CP) is β, and the number of inferior particles in the inferior particle swarm (BP) is γ, and the selection probability of superior particles is determined through the design of a weight function, which is shown in formula (11).

$$W_j = \frac{1}{\sqrt{2\pi\alpha}} e^{-\frac{\text{rank}(j)-1}{2\alpha}} \tag{11}$$

where W_j is the weight of the jth particle in the quality particle swarm, and $rank(j)$ is the position of particle j in the quality particle swarm cluster after sorting according to the fitness value. After obtaining the weights of the quality-particle population, the probability of each particle being selected for learning is derived according to the following equation as shown in formula (12).

$$\rho_j = \frac{w_j}{\sum_{i=1}^{\alpha} w_i} \tag{12}$$

The poor quality particle swarm will select the corresponding high quality particle position for iteration according to the high quality particle probability selection formula, and the update formula for quality particles is shown in formula (13) and formula (14) as follows.

$$v_i^{t+1} = \omega v_i^t + c_2(GP_{randx}^t - x_i^t) \tag{13}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{14}$$

where v_i^{t+1} is the velocity of the ith particle in the poor-quality particle swarm at the $t+1$th iteration, v_i^t is the velocity of the ith particle in the poor-quality particle swarm at the tth iteration, and GP_{randx}^t is the position of the random particles in the high-quality particle swarm.

Due to their poor search performance, low-quality particles will exclude their historical optimal values from the updated formula. Each particle will randomly learn from the particles in the high-quality particle swarm to improve search results. This method can also improve sample diversity by using different learning samples in different search directions.

High-quality particles have excellent search performance and should focus on enhancing local exploration while avoiding local optima. This study introduces a high-quality particle evolution scheme in which each particle compares its fitness with the companions of the randomly selected particles during the iterative process. If performance is excellent, use the formula (15) and formula (16) to update; Otherwise, evolve according to formula (6).

$$v_i^{t+1} = \omega v_i^t + c_2\left(GP_{ix}^t - GP_{rand,rand\neq i}^t\right) \tag{15}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{16}$$

where GP_{ix}^t is the ith quality particle position in the quality particle swarm, and $GP_{rand,rand\neq i}^t$ is the random particle position of the non-particle i in the quality particle swarm.

High-quality particles learn from superior peers, directing them toward optimal regions to enhance search efficiency. Random peer comparisons preserve lower-fitness particles in the swarm, maintaining diversity and preventing local optima entrapment.

In standard PSO, particles learn both from the global best samples and from elite particles, i.e., those particles that show a consistent fitness improvement over the course of the iterations, which indicate an effective search direction. The elite particle population (EP) aggregates such high-potential particles. When EP exists, ordinary particles perform crossover operations by randomly selecting partners from the optimal and elite groups in each iteration. This produces hybrid learning samples that can have both the high-quality positional features of the premium particle swarm and the efficient search features of the elite particle swarm. The update formula for ordinary particles is shown in formula (17) and formula (18):

$$v_{id}^{t+1} = \omega v_{id}^t + c_1 r_1(p_{id}^t - x_{id}^t) + c_2 r_2(m_d^t - x_i^t) \tag{17}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{18}$$

where m_d^t is the position of the crossed particles in the d dimension, and the crossed particles are generated by random crossings.

$$m_d = \begin{cases} EP_d, & \text{if } r > 0.5 \\ GP_d, & \text{otherwise} \end{cases} \tag{19}$$

where r is a random number within [0,1], it can be concluded from formula (19) that the constructed crossover particle samples have the same probability of learning from the quality particle samples and the elite particle samples, which is conducive to constructing fair crossover particles.

3.3 Mixed mutation mechanism

Nature-inspired algorithms often struggle with premature convergence to local optima due to inadequate global exploration. The Cauchy mutation operator addresses this through its broad-tailed distribution, enhancing search space coverage. Its characteristic midpoint-localized small steps and large exploratory jumps disrupt particle stagnation, forcing positional adjustments to escape local traps. The distribution's flat peak and gradual decay reduce post-mutation local search duration, prioritizing global optimization while boosting self-regulation. This strategy balances exploration-exploitation trade-offs and enriches population diversity, ultimately accelerating convergence. The standard Cauchy distribution formula (20) is as follows:

$$f(x) = \frac{1}{\pi}\left(\frac{1}{x^2+1}\right) \tag{20}$$

During the particle swarm optimization process, when the swarm population's global best solution exhibits persistent non-update events over successive optimization cycles, this pattern serves as an empirical indicator of potential convergence to suboptimal regions. To counteract this search stagnation, we strategically apply Cauchy mutation operators to stochastically perturb particle positions, thereby introducing directional diversity within the solution space and probabilistically enabling barrier escape from local attraction basins. The formula for the mutation is shown in formula (21).

$$p_i(t+1) = p_i(t) + \sigma \text{Cauchy}(\theta, \alpha) \tag{21}$$

where $\text{Cauchy}(\theta, \alpha)$ is the Cauchy distribution and σ is a self-set parameter.

4 AHPSO-PPO Algorithm

The PPO framework demonstrates enhanced capacity for rapid neighborhood exploration and high-throughput sample collection during policy iteration. However, under high-dimensional state distributions exhibiting inherent sparsity patterns, this operational characteristic renders the learning system vulnerable to premature convergence in parameter subspaces. Therefore, the problems inherent in the algorithm can be effectively addressed by using the AHPSO algorithm to initialize multiple policy networks, create a population of policy networks interacting with the environment, generate sample data, obtain an optimized policy network, and update the evaluation network based on empirical data. This allows them to learn more efficiently and make policy improvements using a particle swarm algorithm called the AHPSO-PPO algorithm. The structure diagram of AHPSO-PPO algorithm is as in Fig. 3. The pseudo-code for the AHPSO-PPO algorithm is shown in Algorithm 1.

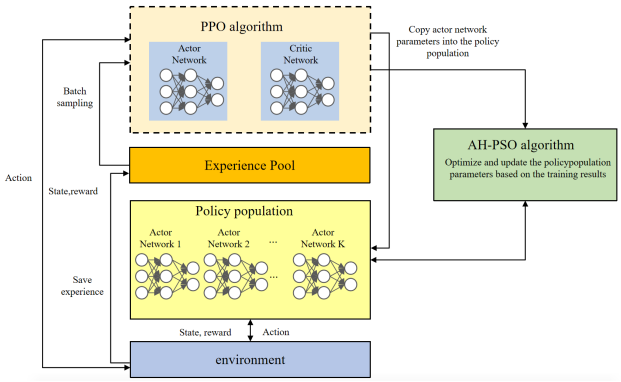


Fig. 3: The structure diagram of AHPSO-PPO algorithm

Algorithm 1 AHPSO-PPO algorithm

for *iteration = 1, 2, …* **do**
  for *episode = 1, 2, …, N* **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
  if *Meet the conditions for AHPSO intervention* **then**
    Assigning PPO network parameters to particles
    $stagnation \leftarrow 0$
    for $t \leftarrow 1$ **to** T **do**
      $w(t) \leftarrow w_{\max} - (w_{\max} - w_{\min})\frac{t}{T}$
      $c_1(t) \leftarrow c_1^{\max} - (c_1^{\max} - c_1^{\min})\frac{t}{T}$
      $c_2(t) \leftarrow c_2^{\min} + (c_2^{\max} - c_2^{\min})\frac{t}{T}$
      Sort the particles in ascending order according to $f(x_i)$, classify the top 20% as the high quality group GP, the middle 60% as the normal group CP, and the bottom 20% as the poor quality group BP
      foreach *particle group* $G \in \{GP, CP, BP\}$ **do**
        foreach *particle* $i \in G$ **do**
          if $i \in GP$ **then**
            $v_i^{t+1} \leftarrow \omega v_i^t +$
            $c_2\left(GP_{ix}^t - GP_{rand,rand\neq i}^t\right)$
          else if $i \in CP$ **then**
            $v_{id}^{t+1} \leftarrow \omega v_{id}^t + c_1 r_1(p_{id}^t - x_{id}^t) + c_2 r_2(m_d^t - x_i^t)$
          else
            $v_i^{t+1} \leftarrow \omega v_i^t + c_2(GP_{randx}^t - x_i^t)$
          end
          $x_i^{t+1} = x_i^t + v_i^{t+1}$
        end
      end
      foreach *particle* i **do**
        if $f(x_i) < f(P_{best_i})$ **then**
          $P_{best_i} \leftarrow x_i$
          if $f(P_{best_i}) < f(G_{best})$ **then**
            $G_{best} \leftarrow P_{best_i}$
            $stagnation \leftarrow 0$
          end
        end
      end
      $stagnation \leftarrow stagnation + 1$
      if $stagnation \geq K$ **then**
        foreach *particle* i **do**
          $p_i(t+1) \leftarrow p_i(t) + \sigma \text{Cauchy}(\theta, \alpha)$
        end
        $stagnation \leftarrow 0$
      end
    end
    $\theta \leftarrow G_{\text{best}}$
  end
end

5 Experiments and Analysis of Results

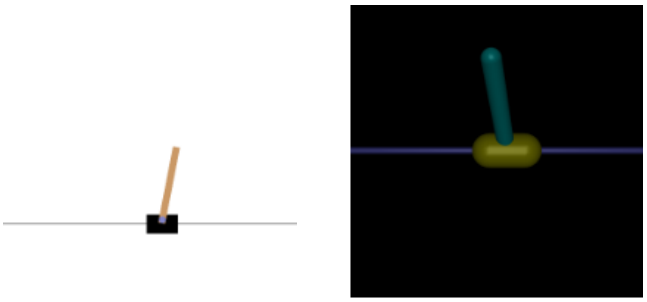
5.1 Experimental Environment

To evaluate the performance of the AHPSO-PPO algorithm, we tested it in a set of controlled environments at the OpenAI Gym.

The cart-pole environment is a classical control problem in reinforcement learning, as shown in Fig. 4a. The objective is to maintain the balance of the connected poles by controlling the cart to move left and right. The state space of this environment consists of four continuous variables: the position and velocity of the cart, the angle of the pole, and the angular velocity. The action space is typically two discrete actions: applying force to the left or applying force to the right. The goal of the intelligent body is to learn a strategy

to keep the pole upright for as long as possible by choosing the action at the right time. The reward mechanism is adding 1 bonus per step for maintaining balance, with the current round ending when the pole falls or the cart goes off track.

The inverted pendulum environment is a classical control problem in reinforcement learning, as shown in Fig. 4b. The aim is to maintain the balance of the inverted pendulum by controlling the platform that moves at the bottom. The system consists of a vertical pendulum and a horizontally movable trolley to which the pendulum is attached via a hinge, and the goal is to control the motion of the trolley so that the pendulum remains in an upright state. The state space consists of four continuous variables: the position of the cart, the velocity, the angle of the pendulum, and the angular velocity. The action space is continuous values representing the magnitude and direction of the force applied to the cart, in the range [-3, 3]. The reward mechanism is set based on how far the pendulum angle deviates and whether the position of the cart is out of range, with the round ending when it falls over or is out of the specified range.



(a) Cart-Pole (b) Inverted Pendulum

Fig. 4: Environments

5.2 Experimental parameter setting

For the DDPG used in our comparison experiments, the policy network architecture is set according to previous literature[28], two hidden layers of a fully connected network, each using tanh nonlinearity. For the AHPSO-PPO proposed in this paper and the PPO used for comparison, they are set up in the same way. We typically construct a forward strategy network with only one hidden layer, moving and adding in the output layer using the tanh activation function, outputting the action.

In the hyperparameter setting of AHPSO, we follow the following principles: a) complex tasks require stronger global search capability, so the population size is appropriately increased; b) the effect of the parameters on the convergence speed and stability is observed through preexperimentation; and c) the range of empirical values is initialised with reference to the classical PSO studies. The specific parameters are shown in Tables 1.

5.3 Results and discussion

The AHPSO-PPO algorithm demonstrates superior performance over PPO and DDPG in the CartPole task(Fig. 5). The horizontal axis in the picture represents the number of training epochs, and the vertical axis represents the average return. The AHPSO-PPO(red curve) exceeded the average return of 400 in approximately 150 rounds, while the PPO(blue curve) required approximately 300 rounds to reach a similar level, and DDPG(blue curve) showed a significant lag in convergence in the early stages, only beginning to steadily rise after 400 rounds. This indicates that the convergence speed of AHPSO-PPO has been improved, and the return fluctuation during the rapid rise phase is smaller, indicating that its parameter update strategy has stronger robustness. These three algorithms ultimately approach the theoretical maximum value of the environment(500), but AHPSO-PPO enters a stable saturation stage around 200 rounds. This enhanced convergence and robustness stem from integrating PSO's global search with PPO's policy gradients, overcoming traditional algorithms' parameter sensitivity and local optima limitations, validating hybrid optimization's potential in discrete control.

Table 1: AHPSO Parameters

Particles	20
Optimization intervals	5
Iterations	20
$w_{\max}, w_{\min}$	0.9, 0.4
$c_1^{\max}, c_1^{\min}$	2.5, 0.5
$c_2^{\max}, c_2^{\min}$	2.5, 0.5
σ	0.1

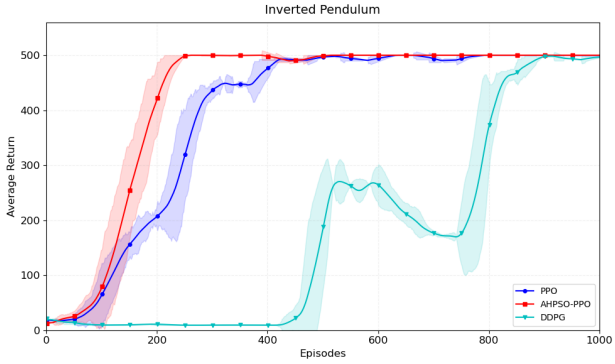


Fig. 5: Simulation results in CartPole

The experimental results show that in the inverted pendulum control task, the AHPSO-PPO algorithm(red curve) outperforms PPO(blue) and DDPG(blue) in terms of convergence efficiency and stability (Fig. 6). The convergence speed of AHPSO-PPO is significantly faster than that of PPO and DDPG. During the rapid rise period, compared to the high volatility of DDPG, the narrow oscillation range (shadow) of AHPSO-PPO reflects its robust parameter update strategy. All algorithms are close to the theoretical maximum return of the environment(500), but AHPSO-PPO stabilizes near saturation after around 200 attacks, much earlier than other algorithms. This enhancement stems from the combination of particle swarm optimization(PSO) and PPO strategy gradient, which reduces the sensitivity of parameters and local convergence traps in traditional methods. The results highlight the effectiveness of the hybrid optimization framework in discrete control systems.

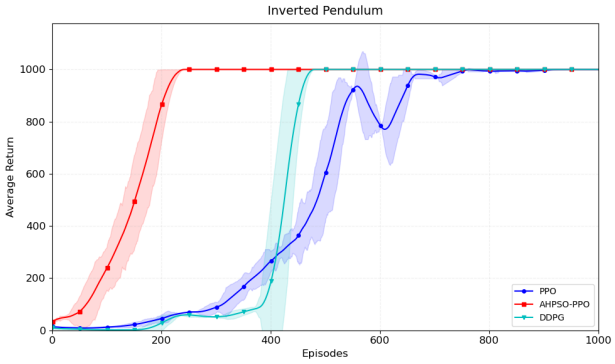


Fig. 6: Simulation results in Inverted Pendulum

6 Conclusion

This study proposes the AHPSO-PPO framework to address reinforcement learning challenges like slow convergence and premature suboptimal policies. It integrates adaptive parameter tuning, hierarchical particle swarm optimization, and variational evolution strategies, optimizing PPO network parameters through swarm intelligence. Experimental results demonstrate superior performance over standard PPO and DDPG, with accelerated convergence and enhanced environmental robustness. The hybrid approach bridges gradient-based policy optimization and population-based search, offering a scalable solution for real-world decision systems that require precision and adaptability. Future research will focus on two main directions: first, applying the AHPSO-PPO framework to complex continuous control tasks to verify its generalization ability; second, developing lightweight algorithms to reduce computational costs.

References

[1] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[2] Michael I Jordan and Tom M Mitchell. "Machine learning: Trends, perspectives, and prospects". In: *Science* 349.6245 (2015), pp. 255–260.

[3] Martin L Puterman. "Markov decision processes". In: *Handbooks in operations research and management science* 2 (1990), pp. 331–434.

[4] Marcin Andrychowicz et al. "Learning to learn by gradient descent by gradient descent". In: *Advances in neural information processing systems* 29 (2016).

[5] Tanmay Gangwani and Jian Peng. "Policy optimization by genetic distillation". In: *arXiv preprint arXiv:1711.01012* (2017).

[6] Xinyue Chen et al. "Randomized ensembled double q-learning: Learning fast without a model". In: *arXiv preprint arXiv:2101.05982* (2021).

[7] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *nature* 575.7782 (2019), pp. 350–354.

[8] Chen Tang et al. "Deep reinforcement learning for robotics: A survey of real-world successes". In: *Annual Review of Control, Robotics, and Autonomous Systems* 8 (2024).

[9] Arta Iftikhar et al. "A reinforcement learning recommender system using bi-clustering and Markov Decision Process". In: *Expert Systems with Applications* 237 (2024), p. 121541.

[10] Hongxia Cai, Yunqi Bian, and Lilan Liu. "Deep reinforcement learning for solving resource constrained project scheduling problems with resource disruptions". In: *Robotics and Computer-Integrated Manufacturing* 85 (2024), p. 102628.

[11] Claire Glanois et al. "A survey on interpretable reinforcement learning". In: *Machine Learning* 113.8 (2024), pp. 5847–5890.

[12] Akifumi Wachi, Xun Shen, and Yanan Sui. "A survey of constraint formulations in safe reinforcement learning". In: *arXiv preprint arXiv:2402.02025* (2024).

[13] Olivier Sigaud. "Combining evolution and deep reinforcement learning for policy search: A survey". In: *ACM Transactions on Evolutionary Learning* 3.3 (2023), pp. 1–20.

[14] Evgenii Nikishin et al. "The primacy bias in deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2022, pp. 16828–16847.

[15] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PmLR. 2016, pp. 1928–1937.

[16] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[17] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. ieee. 1995, pp. 1942–1948.

[18] Federico Marini and Beata Walczak. "Particle swarm optimization (PSO). A tutorial". In: *Chemometrics and Intelligent Laboratory Systems* 149 (2015), pp. 153–165.

[19] Daniel Ocran, Sunday Sunday Ikiensikimama, and Eric Broni-Bediako. "A compositional function hybridization of PSO and GWO for solving well placement optimisation problem". In: *Petroleum Research* 7.3 (2022), pp. 401–408.

[20] Thanh Cuong-Le et al. "An efficient approach for damage identification based on improved machine learning using PSO-SVM". In: *Engineering with Computers* (2022), pp. 1–16.

[21] Baoye Song, Zidong Wang, and Lei Zou. "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve". In: *Applied soft computing* 100 (2021), p. 106960.

[22] Jie Zhou et al. "Leakage diagnosis and localization of the gas extraction pipeline based on SA-PSO BP neural network". In: *Reliability Engineering & System Safety* 232 (2023), p. 109051.

[23] Feng Wang, Xujie Wang, and Shilei Sun. "A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization". In: *Information Sciences* 602 (2022), pp. 298–312.

[24] Arpita Choudhary, NC Barwar, and Vikas Chouhan. "Energy efficient cluster head selection using hybrid RL-PSO approach". In: *Full Length Artic* 11.2 (2024), pp. 08–8.

[25] Shifei Ding et al. "A new asynchronous reinforcement learning algorithm based on improved parallel PSO". In: *Applied intelligence* 49 (2019), pp. 4211–4222.

[26] Tundong Liu et al. "A novel policy gradient algorithm with PSO-based parameter exploration for continuous control". In: *Engineering Applications of Artificial Intelligence* 90 (2020), p. 103525.

[27] Yaozhong Zhang et al. "Multi-UAV pursuit-evasion gaming based on PSO-M3DDPG schemes". In: *Complex & Intelligent Systems* 10.5 (2024), pp. 6867–6883.

[28] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).