2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Adaptive Parameter Control in Particle Swarm Optimization Based on Proximal Policy Optimization

Ruiqi Fan[1[0009-0002-4230-665X]] and Lisong Wang[1[0000-0001-6482-3717]] and Shaohan Liu[1[0000-0003-4290-6591]] Liang Liu[1[0000-0002-4903-2666]] and Fengtao Xu[2[0009-0009-6308-4946]] and Yizhuo Sun[2[0009-0000-7755-8561]]

[1] Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[2] China Academy of Launch Vehicle Technology, Beijing 100076, China
wangls@nuaa.edu.cn

Abstract. The performance of Particle Swarm Optimization algorithms when solving complex problems is highly dependent on parameter settings and prone to premature convergence. This paper proposes a Reinforcement Learning-based adaptive multi-subgroup PSO framework, termed PPOPSO, designed to enable an RL agent to learn online and dynamically adjust the core behavioral parameters of multiple parallel PSO subgroups.This framework utilizes Proximal Policy Optimization as the RL agent. Its decision-making process is informed by a state representation that incorporates rich historical performance indicators. Furthermore, it independently selects actions for each subgroup from a predefined set of parameter configurations, each representing different search strategies. The controlled multi-subgroup PSO system integrates a periodic elite particle migration mechanism to foster information sharing and maintain diversity among the subgroups.This design transforms the parameter adaptation challenge into a sequential decision-making process. This allows the system to autonomously balance exploration and exploitation based on the optimization stage and the state of each subgroup. Preliminary experimental results on the CEC2013 standard test function set indicate that, through dynamic parameter adjustment empowered by reinforcement learning, the proposed PPOPSO framework can exhibit superior performance compared to traditional methods, offering a promising new approach for complex optimization problems.

Keywords: Particle Swarm Optimization , Reinforcement Learning , Proximal Policy Optimization.

1 Introduction

Evolutionary algorithms represent a class of typical gradient-free black-box optimization methods[1]. Based on Darwin's theory of evolution, they iteratively evolve solutions. Over the past few decades, evolutionary algorithms have become a research hotspot in both academia and industry due to their demonstrated high efficiency and robustness in solving complex optimization problems. They have provided powerful

optimization tools for numerous fields, including material structure design[16], robot path planning[10], and multi-objective scheduling[18].

Among these algorithms, Particle Swarm Optimization (PSO)[7] is widely used due to its few parameters, simple structure, and ease of implementation. Proposed by Kennedy and Eberhart in 1995, PSO was inspired by social behaviors such as bird flocking and fish schooling, and explores the problem space by updating the velocity and position of particles. However, research indicates that as the dimensionality of optimization problems increases, PSO suffers from drawbacks such as premature convergence and poor convergence performance. For instance, it is prone to converging to local optima too early, and the population diversity rapidly decreases in the later stages of the search, thereby affecting its global search capability[19].To overcome these shortcomings, many methods have been proposed to adaptively set particle swarm parameters for improvement[15]. For example, the literature[17] designed two mutation operators to enhance the exploration ability of the particle swarm. However, these methods often require expert experience to design control rules, which limits their adaptability.

In recent years, Reinforcement Learning (RL)[14], with its ability to learn through interaction with the environment in complex decision-making problems, has provided a new perspective for the adaptive control of optimization algorithm parameters. Compared with traditional adaptive methods, RL can autonomously learn control strategies, reducing the dependence on prior knowledge[8]. Inspired by this, we propose an adaptive parameter control framework based on reinforcement learning, which uses the PPO algorithm. By observing the running state of PSO[13], it selects appropriate parameter configurations for multiple parallel evolving subpopulations (particle swarms), aiming to balance global exploration and local exploitation capabilities in different optimization stages, and to promote effective information sharing and migration among subpopulations.

The main contributions of this paper can be summarized as follows:

1. We propose a novel multi-subgroup PSO parameter dynamic control framework based on Proximal Policy Optimization. This framework allows the RL agent to independently and adaptively select and switch the core control parameters of PSO for multiple parallel evolving subgroups based on real-time feedback of the optimization state, rather than relying on fixed or predefined parameter decay strategies.
2. We design a reinforcement learning state representation method that includes rich historical performance indicators. The state vector not only includes the current usage of function evaluation resources but also incorporates past historical information. This temporal state representation provides the agent with deep insights into search trends and the recent performance of each subgroup, helping it to learn longer-term control strategies.
3. We divide the particle swarm into multiple subgroups and construct an integrated periodic elite migration strategy. This migration mechanism aims to promote knowledge sharing, maintain population diversity, and synergize with the dynamic parameter regulation of the RL agent to jointly enhance the algorithm's global search capability and convergence performance.

These contributions collectively constitute a complete PSO system that can self-optimize through learning, providing a promising new approach for solving complex optimization problems. Furthermore, through experimental verification on the CEC2013 standard test function set, we demonstrate the potential and effectiveness of the proposed RL control strategy compared to fixed-parameter PSO and some other adaptive methods.

The remainder of this paper is organized as follows: Section 2, Related Work, introduces particle swarm optimization and reinforcement learning. Section 3 introduces a multi-particle swarm optimization based on Proximal Policy Optimization. Section 4 discusses and analyzes the experimental results. Section 5 summarizes the conclusions and proposes directions for future work.

2 Related Work

2.1 Particle Swarm Optimization

PSO[7] explores the search space through a swarm of particles. Each particle represents a potential solution to the optimization problem and adjusts its flight trajectory based on its own experience and the experience of other particles in the swarm.

In the PSO algorithm, each particle i in the swarm has its current position x_i and current velocity v_i in the D-dimensional search space. Simultaneously, each particle records the best position it has visited so far, known as the personal best solution $pbest_i$. Furthermore, the best position found by the entire swarm so far is tracked as the global best solution $gbest$. The movement of each particle is influenced by three factors: its current velocity, its personal best solution, and the global best solution.

The core of the PSO algorithm lies in iteratively updating the velocity and position of each particle. In each iteration t, the velocity of particle i in the d-th dimension is updated according to the following formula:

$$v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_1 \cdot \left(pbest_{id}(t) - x_{id}(t)\right) + c_2 \cdot r_2 \cdot \left(gbest_d(t) - x_{id}(t)\right) \quad (1)$$

where w is the inertia weight, used to control the influence of the previous velocity on the current velocity, thereby balancing global exploration and local exploitation capabilities. c_1 and c_2 are acceleration constants, representing the cognitive learning factor and the social learning factor, respectively. r_1 and r_2 are random numbers uniformly distributed in the range [0, 1]. $pbest_{id}(t)$ is the personal best position of particle i in the d-th dimension as of iteration t, and $gbest_d(t)$ is the global best position found by the entire swarm in the d-th dimension as of iteration t.

Subsequently, the position of particle i in the d-th dimension is updated based on its updated velocity:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

This iterative process continues until a preset stopping condition is met, such as reaching the maximum number of iterations or obtaining a sufficiently good fitness value.

2.2 Reinforcement Learning

Reinforcement Learning (RL)[14] is a machine learning method where an agent learns an optimal policy through interaction with an environment. In reinforcement learning, after taking an action a in a specific state s, the agent receives a reward r from the environment and transitions to a new state s'. Its objective is to find a policy $\pi(a|s)$ that maximizes the cumulative reward obtained by the agent over the long term. The policy in reinforcement learning can be updated based on value functions. Common value functions include the state-value function $V^{\pi}(s)$ and the state-action value function $Q^{\pi}(s,a)$, which represent the expected cumulative reward when following policy π in state s, and the expected cumulative reward when taking action a in state s and then following policy π, respectively. Typically, the value function update formula is the Bellman expectation equation:

$$Q^{\pi}(s,a) = E[r + \gamma V^{\pi}(s')|s,a] \tag{3}$$

where γ is the discount factor, used to balance the weight of current rewards and future rewards. The policy can be updated based on the value function, for example, using policy gradient methods, where the update formula is:

$$\nabla_{\theta} J(\theta) = E_{s,a}[\nabla_{\theta} \ln \pi_{\theta}(a|s) Q^{\pi}(s,a)] \tag{4}$$

By continuously iterating through policy and value function updates, reinforcement learning enables the agent to gradually learn the optimal behavioral policy in the environment to maximize long-term cumulative rewards.

3 Methodology

3.1 The PPOPSO Framework

The novel optimization algorithm proposed in this paper, named PPOPSO, combines Proximal Policy Optimization (PPO) with multi-subgroup Particle Swarm Optimization , aiming to achieve dynamic adaptive control of PSO's core parameters. Firstly, a random initialization method is employed to initialize PPO's policy network π_{θ} and the multiple subgroups of PSO particles. Secondly, action selection is performed by the PPO policy network based on the current state s_t, which predicts an optimal operator configuration a_t for each subgroup. Thirdly, according to the actions selected by the PPO agent, each subgroup executes the corresponding particle update operations and evaluates their fitness. Then, the historical experience (s_t, a_t, r_t, s_{t+1}) generated during this decision cycle is recorded in the experience buffer $\mathcal{B}$. Subsequently, the PPO policy network parameters are updated based on the data in the experience buffer. The pseudocode for PPOPSO is shown in Algorithm 1.

Algorithm 1 PPOPSO Algorithm:

Input: Fitness function $f(\cdot)$, Problem dimension D, Maximum number of function evaluations FES_{max}

Output: Best fitness value of the population G_{best_cost}

1: Initialize multi-subgroup particles (positions, velocities) and basic PSO parameters; Initialize RL policy network π_θ.

2: $FES_t \leftarrow 0$.

3: while $FES_t < FES_{max}$ **do**

4: Calculate state s_t according to Equation(5).

5: Obtain action a_t based on policy network π_θ and state s_t using Equation(9).

6: **for** each subgroup $j \leftarrow 1$ to N_s **do**

7: Update particle velocity and position using $a_t[j]$ according to Equations(1) and(2).

8: Update $pbest$ for all particles, $gbest_j$ for each subgroup, and global C^t_{gbest}.

9: **if** $iter \,\%\, migration\_interval = 0$

10: Perform particle migration.

11: Calculate reward r_t based on global C^t_{gbest} using Equations(7) and (8).

12: Calculate next state s_{t+1} according to Equation(5).

13: $done \leftarrow$ Check if $FES_t \geq FES_{max}$.

14: Store $(s_t, a_t, r_t, s_{t+1}, done)$ in buffer $\mathcal{B}$.

15: $s_t \leftarrow s_{t+1}$.

16: **if** length$(\mathcal{B}) \geq N_{collect}$ or $done$:

17: Perform PPO Update {Train π_θ and V_θ networks}.

18: Clear buffer $\mathcal{B}$.

19: Return G_{best_cost}.

3.2 State Design

The state vector S_t at decision step t is a concatenation of several key pieces of information. It can be represented as:

$$S_t = [FE_{\text{norm}}(t), C^{t-1}_{gbest}, \bar{C}^{t-1}_{pbest,1}, \dots, \bar{C}^{t-1}_{pbest,N_s}, \dots, C^{t-H}_{gbest}, \bar{C}^{t-H}_{pbest,1}, \dots, \bar{C}^{t-H}_{pbest,N_s}] \quad (5)$$

Let H be the history length and N_s be the number of subgroups. Metrics from the k RL steps preceding the current step t are denoted with the superscript $(t-k)$.

1. **Normalized Function Evaluations ($\boldsymbol{FEs_{\text{norm}}}$)**: This scalar value represents the proportion of the total optimization budget (Max_FEs) that has been consumed. It is calculated as $FEs_{norm}(t) = \frac{FEs_t}{FEs_{max}}$, where FEs_t is the number of function evaluations consumed up to RL step t, and FEs_{max} is the maximum allowed number of function evaluations. This component helps the agent understand its current position on the optimization timeline.
2. **Historical Performance Metrics**: To provide the agent with an understanding of performance trends, the state includes a history of key metrics over a defined window of H previous RL steps (where H is history_len). For each of these H steps, two types of metrics are recorded:
   - **Global Best Cost ($\boldsymbol{C_{gbest}}$)**: The global best fitness value found by any particle across all swarms up to that historical point.
   - **Average Personal Best Cost per Swarm ($\boldsymbol{\overline{C}_{pbest,j}}$)}**: For each of the N_s swarms (where N_s is num_swarms), this is the average of the personal best fitness values of all particles within that swarm j.

During initialization, the historical components are populated with a predefined large value to represent a poor initial state.

3.3 Action Design

The agent's action is the primary means by which it interacts with and exerts control over the Particle Swarm Optimization process. The design of the action directly influences the agent's ability to adjust the PSO's behavioral strategy, thereby affecting overall optimization performance.

The action space includes five action configurations as shown in the table. Each configuration contains specific PSO parameters, such as the inertia weight w, the cognitive learning factor c_1, and the social learning factor c_2. Table 1 details these five configurations and their specific parameter values.

Table 1. Predefined PSO Parameter Configuration Options

w	c_1	c_2	Other Parameters	Description
0.729	1.49445	1.49445	-	Standard PSO parameters
0.9	1.8	1.2	-	Strong Exploration
0.4	1.2	1.8	-	Strong Exploitation
-	1.5	1.5	$w_{range} = (0.9,0.4)$	Linearly Decreasing Inertia Weight
0.6	2.0	1.0	-	Another parameter variant

These configurations can cover multiple strategies, allowing the agent to assign appropriate exploration or exploitation strategies to each particle swarm based on its current state and historical performance.

The agent selects a specific PSO parameter configuration for each independent particle swarm in the environment. At each decision step t, the action A_t selected by the agent can be represented as a vector containing N_s elements:

$$A_t = [idx_0, idx_1, \dots, idx_{N_s-1}] \tag{6}$$

where idx_j is an integer index representing the number of the predefined parameter configuration selected for the j-th particle swarm.

By selecting from these preset configurations, after the action is executed, the chosen configuration will guide each particle swarm to operate according to the parameters of that action configuration. This mechanism enables the RL agent to periodically and specifically adjust the operating parameters of the multi-swarm PSO to adapt to the characteristics of the optimization problem at different stages.

3.4 Reward Design

The global best cost is denoted as C^t_{gbest}. The global best cost at the end of the previous RL decision step $t-1$ is denoted as C^{t-1}_{gbest}. The improvement in cost, ΔC_t, is calculated as the difference between these two:

$$\Delta C_t = C^{t-1}_{gbest} - C^t_{gbest} \tag{7}$$

A concise design is adopted to evaluate the immediate effect of the action selected by the agent at step t. The specific rules for assigning the reward R_t are as follows:

$$R_t = \begin{cases} +1.0, & \text{if } \Delta C_t > 0 \\ -0.1, & \text{if } \Delta C_t = 0 \end{cases} \tag{8}$$

This penalty mechanism aims to encourage the agent to avoid selecting parameter configurations that lead to algorithm stagnation or performance degradation.

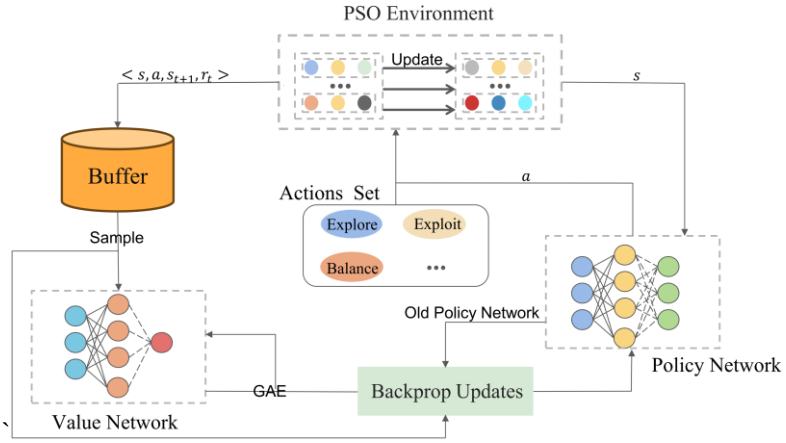

Fig. 1. The PPOPSO Framework Workflow.

3.5 Training Process

As shown in
Fig. **1**, in the reinforcement learning-controlled PSO framework proposed in this paper, the PPO algorithm is used to train an agent. This agent dynamically selects the optimal core control parameter configurations based on the real-time state of the PSO algorithm. The training process of PPO is an iterative online learning process, where the agent collects experience by interacting with the PSO environment and uses this experience to update its policy network and value function network.

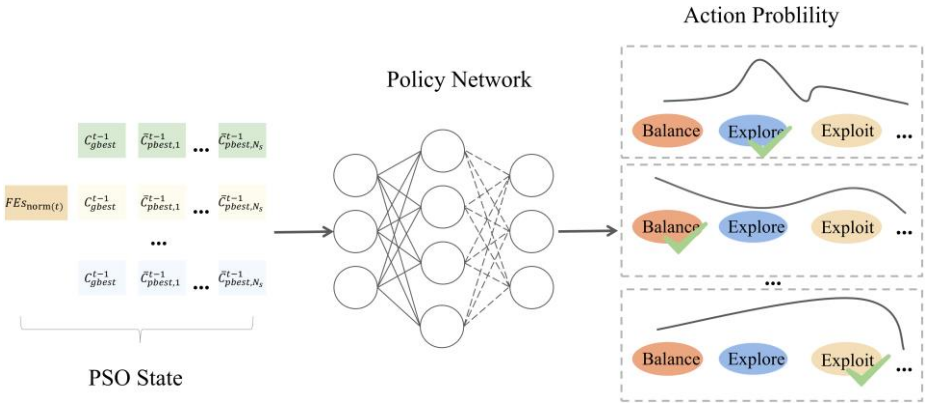

Fig. 2. Action Selection Mechanism via Policy Network.

At each RL decision time step t, the agent first observes the current state s_t from the PSO environment. As shown in Fig. 2, after the policy network receives the state s_t, its output layer generates a corresponding "logit" value for each subgroup j and each possible parameter configuration index k, denoted as $L_{j,k}(s_t;\theta)$. These logits are then independently transformed into probability distributions for each subgroup j using the Softmax function.

The probability of selecting the k-th parameter configuration for the j-th subgroup, $P(a_{t,j}=k|s_t;\theta)$, is calculated as follows:

$$P(a_{t,j}=k|s_t;\theta)=\frac{\exp\left(L_{j,k}(s_t;\theta)\right)}{\sum_{k'=0}^{4}\exp\left(L_{j,k'}(s_t;\theta)\right)} \tag{9}$$

After the PSO completes its evolution, the RL environment evaluates its performance. Firstly, the environment returns an immediate reward r_t, based on the improvement in C_{best} compared to the best cost at the end of the previous RL step. Simultaneously, the environment generates and returns the next state s_{t+1}. This complete interaction data, i.e., the experience tuple (s_t, a_t, r_t, s_{t+1}), along with the log probability of the action a_t under the old policy $\log \pi_{\theta_{\text{old}}}(a_t|s_t)$, are collected together. This serves as the basis for the subsequent PPO algorithm to update the policy network parameters θ.

Concurrently, a value function network $V_\phi(s_t)$ works in coordination. The goal of this network is to learn to estimate the the state value $V(s_t)$ that can be obtained by following the current policy given the PSO state s_t, where ϕ represents the parameters of the value function network.

During the training process, the agent first uses the current policy $\pi_{\theta_{\text{old}}}$ to interact with the PSO environment and collect a batch of experience data. Then, using this data, it performs multiple iterative updates on the parameters θ and ϕ through stochastic gradient ascent and stochastic gradient descent, respectively.

3.6 Network Architecture

Both networks employ a Multi-Layer Perceptron structure. Specifically, both the policy network and the value network consist of an input layer, two hidden layers each containing 64 neural units (using Tanh as the activation function between hidden layers), and an output layer. The common input for both networks is the state vector s_t, with a dimension of $1 + H \times (1 + N_s)$, which is specifically determined by the state representation of the PSO environment.

For the policy network (Actor), its output layer is a linear layer with an output dimension of $N_s \times 5$. This output corresponds to the logit values for selecting one action from the 5 predefined PSO parameter configurations for each of the N_s subgroups. These logits are then converted into probabilities for each action via the Softmax function (applied independently to the action selection for each subgroup). For the value network (Critic), its output layer is also a linear layer, outputting a scalar value with a dimension of 1, which is the value estimate $V_\phi(s_t)$ for the current state s_t.

Table 2. PPO Network Architecture

Layer No.	Value Network (Critic Network)		Policy NetWork(Actor Network)	
	Operation	Parameters	Operation	Parameters
1	Linear(FullyConnected)	$in\_features = 1 + H \times (1 + N_s), out\_features = 64$	Linear(FullyConnected)	$in\_features = 1 + H \times (1 + N_s), out\_features = 64$
2	Tanh(Activation)	-	Tanh(Activation)	-
3	Linear(FullyConnected)	$in\_features = 64, out\_features = 64$	Linear(FullyConnected)	$in\_features = 64, out\_features = 64$
4	Tanh(Activation)	-	Tanh(Activation)	-
5	Lineat(ValueOutput)	$in\_features = 64, out\_features = 1$	Linear(PolicyLogits)	$in\_features = 64, out\_features = N_s \times 5$

3.7 Particle Migration

To enhance the global search performance of the proposed multi-subgroup particle swarm optimization algorithm and effectively prevent individual subgroups from prematurely converging to local optima, this study designs and implements an inter-subgroup particle migration mechanism. The core objective of this mechanism is to promote the exchange and sharing of high-quality information among different subgroups.

The particle migration mechanism in this study is triggered at a predefined fixed iteration interval I_m.

The migration process begins by selecting a predetermined number $N_{mig,j}$ of the best individuals from each source subgroup j, based on their personal best fitness values ($pbest\_cost$), to serve as migrating particles. These elite particles then migrate following a unidirectional ring topology, i.e., from the source subgroup j to the target sub-

group $j' = (j + 1)\%N_s$, where N_s is the total number of subgroups. In the target subgroup j', these incoming elite particles replace an equal number of particles with the currently worst personal best fitness values. This "survival of the fittest" replacement strategy helps to improve the overall solution quality of the target subgroup and may introduce new effective search directions.

The velocity vectors of newly migrated particles are completely re-initialized randomly within preset velocity limits. This velocity reset operation aims to eliminate the flight inertia carried over from the particle's original subgroup, thereby enabling it to adapt more quickly to the new subgroup's local search environment and the search dynamics currently guided by the parameters set by the reinforcement learning agent.

4 Experiments

4.1 Experimental Setup

The experiments utilize the CEC 2013 standard test function set for performance evaluation. For each test function, the problem dimension D is set to 30. The total number of particles in the Particle Swarm Optimization is set to 100, which are allocated to $N_s = 4$ subgroups. A periodic elite migration strategy is employed within each subgroup, with a migration interval set to 10 PSO iterations, a migration rate of 0.1, and a velocity clamping factor of 0.5. The maximum number of function evaluations FEs_{max} is set to 2×10^4.

The training of the PPO agent uses the following main hyperparameter settings:

The mini-batch size is 64; the number of PPO optimization epochs is 10; the discount factor γ is 0.99; the GAE parameter λ is 0.95; the PPO clipping range clip_range is 0.2; the entropy coefficient is 0.01 to encourage exploration.

The algorithm performance is primarily evaluated using metrics such as the mean and standard deviation of the best objective function values obtained over multiple independent runs, and the average number of function evaluations required to converge to a specific precision.

All experiments were executed in a specific computational environment. The operating system was Ubuntu 24.04.1; the CPU was an Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz; the GPUs were 8 NVIDIA Tesla V100-PCIE-32GB units; and the system memory was 512GB. The method framework was implemented using Python 3.9 and PyTorch.

4.2 Effectiveness of the involved strategies

To validate the effectiveness of our method, we designed three variants of the approach:

PPOPSO1: Sets the history length H to 1.

PPOPSO2: Sets the population size to 1 and does not use the migration mechanism.

PPOPSO3: Removes the control part of the PPO agent.

These variants aim to isolate and evaluate the impact of the key components in our proposed PPOPSO method: historical information, population diversity and migration,

and PPO agent control. The four methods were run independently 51 times on each function, and the average of these runs was taken as the result.

The rankings of the four algorithms on different test functions are presented in the form of a heatmap. PPOPSO is displayed on the bottom row of the heatmap, showing the most outstanding performance. This demonstrates the effectiveness of our proposed method, indicating that the PPO agent is the core driver of performance. Historical information provides a basis for PPO's decisions, helping to improve performance. Subgroups and migration enhance the algorithm's exploration capabilities and robustness.

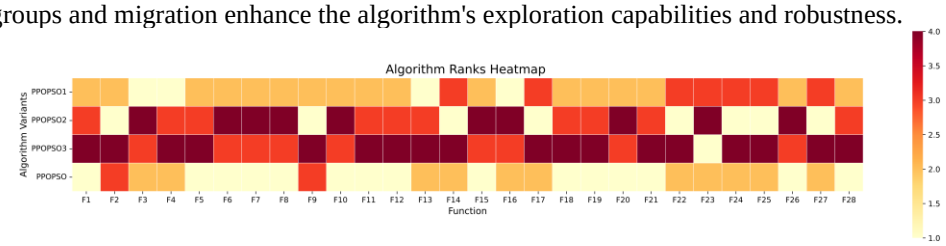

Fig. 3. Alogrithm Ranks Heatmap

Simultaneously, we conducted the Wilcoxon signed-rank test[4] for statistical analysis. The results of the Wilcoxon signed-rank test are shown in Table 3. Specifically, all calculated p-values (7.07E-03, 4.73E-04, and 3.73E-08, respectively) are far below the preset significance levels of $\alpha = 0.05$ and $\alpha = 0.1$. This clearly indicates that the performance improvement demonstrated by PPOPSO over its three variants is highly statistically significant.

Table 3. Wilcoxon Test Results

PPOPSO vs	R^+	R^-	Z	p-value	$\alpha = 0.05$	$\alpha = 0.1$
PPOPSO1	319.0	87.0	-2.693	7.07E-03	yes	yes
PPOPSO2	349.0	57.0	-3.496	4.73E-04	yes	yes
PPOPSO3	403.0	3.0	-5.503	3.73E-08	yes	yes

Fig. 4 shows the effect of the convergence curves for PPOPSO and its three variants. It can be seen that it achieves the best results on different test functions.

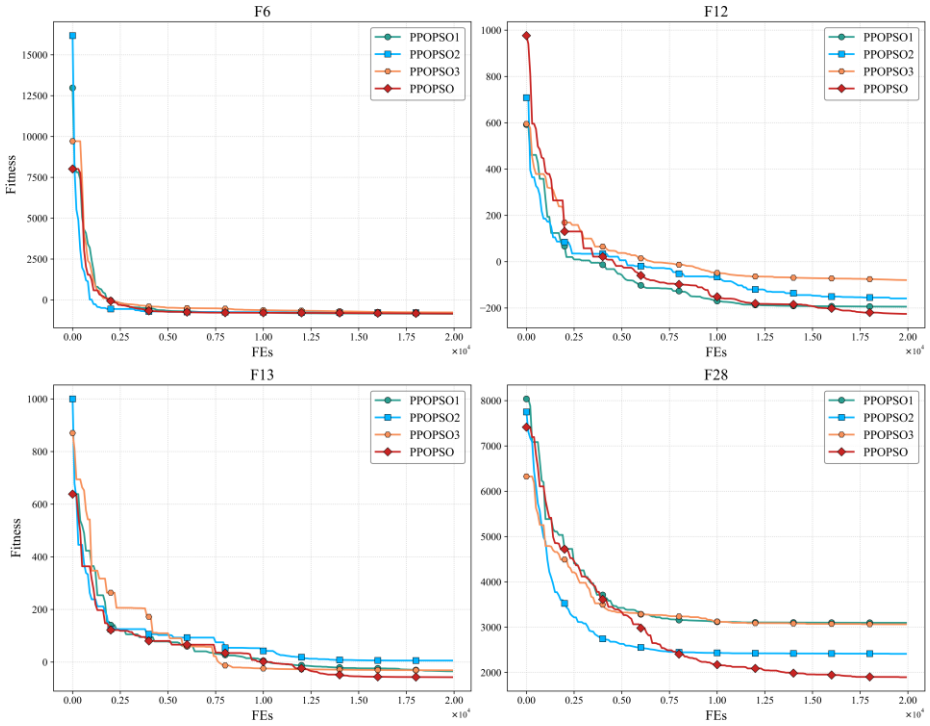

Fig. 4. Convergence curves for PPOPSO and its variants on some functions.

4.3 Performance Comparison

The performance of PPOPSO was evaluated using the CEC 2013[9] benchmark test set. For comparison, six algorithms were employed, including ABC[6], SHPSO[5], QLPSO[11], SOA[2] , GWO-WOA[12] , and OOA[3]. The test results of the comparison algorithms are listed in Table 4 . Each algorithm underwent an equal number of evaluations on each function.

To reduce the impact of experimental errors, each function was run independently 51 times, and the average of these runs was taken as the result. The mean and standard deviation for each function under the 30D dimension were recorded for all comparison algorithms. The best value among all algorithms for each function is highlighted in bold. The bold data represents the best result obtained for each function across all comparison algorithms. The comparison results show that PPOPSO achieved an average rank of 1.78, the highest average rank among all algorithms.

Fig. 5. curves of some functions.shows the convergence curves of PPOPSO against the comparison algorithms on some of the CEC 2013 test functions. It can be clearly observed from the figure that PPOPSO exhibits excellent convergence performance on most functions. This is mainly attributed to the reinforcement learning mechanism, which enables the algorithm to adaptively adjust its strategy based on the current search

state, effectively avoiding the stagnation phenomenon that many other algorithms are prone to.

Fig. 6. Boxplots of some functions.displays the box plots of the final fitness values obtained by each algorithm. It can be seen from the figure that PPOPSO demonstrates high stability on most functions. This stability benefits from the synergistic effect of the PPO agent's learning ability and the population migration mechanism, allowing the algorithm to more reliably converge to high-quality solution regions and effectively reduce the likelihood of getting trapped in local optima.

Table 4. Running results of algorithms in the CEC2013 (30-D).

Function	Metric	ABC	SHPSO	QLPSO	SOA	OOA	GWO-WOA	PPOPSO
F1	Mean	2.90E+03	-1.29E+03	1.10E+02	4.19E+04	2.16E+04	8.84E+01	**-1.34E+03**
	Std	7.87E+02	3.99E+02	1.77E+03	4.38E+03	6.40E+03	7.55E+02	2.23E+02
F2	Mean	3.00E+08	4.90E+07	**1.38E+07**	4.61E+08	1.76E+08	4.43E+07	2.71E+07
	Std	5.62E+07	2.93E+07	7.11E+06	1.64E+08	9.40E+07	1.43E+07	1.09E+07
F3	Mean	5.42E+10	1.94E+10	1.74E+10	1.23E+15	2.83E+12	**6.05E+09**	6.40E+09
	Std	5.20E+09	1.40E+10	1.47E+10	3.32E+15	1.08E+13	2.99E+09	7.17E+09
F4	Mean	1.10E+05	5.04E+04	3.81E+04	5.85E+04	4.58E+04	**3.45E+04**	5.81E+04
	Std	1.16E+04	1.31E+04	1.08E+04	5.11E+03	7.81E+03	9.26E+03	1.57E+04
F5	Mean	1.03E+03	-7.84E+02	-5.22E+02	4.79E+04	1.05E+04	-1.28E+02	**-9.89E+02**
	Std	3.34E+02	3.68E+02	6.55E+02	1.82E+04	4.47E+03	3.10E+02	3.47E+01
F6	Mean	-4.62E+02	-8.02E+02	-7.57E+02	7.01E+03	1.68E+03	-7.26E+02	**-8.32E+02**
	Std	6.74E+01	3.46E+01	9.10E+01	1.66E+03	1.02E+03	6.83E+01	2.90E+01
F7	Mean	1.67E+05	1.47E+05	9.82E+04	1.74E+07	2.36E+06	**8.24E+04**	1.10E+05
	Std	1.08E+04	2.50E+04	2.75E+04	1.45E+07	5.53E+06	1.49E+04	2.83E+04
F8	Mean	-6.79E+02	-6.79E+02	-6.79E+02	-6.79E+02	-6.79E+02	-6.79E+02	**-6.79E+02**
	Std	5.89E-02	4.43E-02	5.40E-02	6.40E-02	4.83E-02	4.87E-02	8.60E-02
F9	Mean	-5.60E+02	-5.68E+02	**-5.79E+02**	-5.63E+02	-5.65E+02	-5.76E+02	-5.72E+02
	Std	1.07E+00	3.03E+00	4.48E+00	1.39E+00	2.13E+00	2.48E+00	3.65E+00
F10	Mean	1.32E+03	-3.23E+02	-2.23E+02	5.48E+03	2.43E+03	-1.06E+02	**-4.30E+02**
	Std	2.60E+02	1.19E+02	2.21E+02	9.15E+02	7.07E+02	1.42E+02	5.99E+01
F11	Mean	-8.29E+01	**-3.07E+02**	-2.88E+02	3.15E+02	1.28E+02	-1.74E+02	-3.01E+02
	Std	1.60E+01	2.18E+01	2.67E+01	5.41E+01	9.55E+01	3.34E+01	3.49E+01
F12	Mean	4.98E+01	-8.44E+01	-1.78E+02	3.62E+02	2.12E+02	-4.23E+01	**-1.80E+02**
	Std	1.96E+01	6.24E+01	3.76E+01	4.40E+01	8.22E+01	2.36E+01	3.44E+01
F13	Mean	1.35E+02	7.62E+01	5.84E-01	4.06E+02	2.80E+02	4.61E+01	**-8.48E+00**
	Std	2.13E+01	3.72E+01	3.33E+01	5.50E+01	8.14E+01	2.11E+01	4.01E+01
F14	Mean	7.56E+03	**2.94E+03**	4.04E+03	7.21E+03	5.85E+03	7.26E+03	3.09E+03
	Std	2.36E+02	5.57E+02	8.95E+02	2.63E+02	6.31E+02	5.76E+02	5.94E+02
F15	Mean	7.98E+03	5.66E+03	7.12E+03	7.76E+03	6.49E+03	7.81E+03	**4.65E+03**
	Std	2.74E+02	6.41E+02	1.05E+03	2.76E+02	7.46E+02	3.75E+02	5.77E+02
F16	Mean	2.03E+02	2.02E+02	2.03E+02	2.03E+02	2.03E+02	2.03E+02	**2.01E+02**
	Std	3.35E-01	5.95E-01	4.06E-01	4.09E-01	4.29E-01	4.29E-01	5.84E-01
F17	Mean	8.33E+02	4.32E+02	**4.13E+02**	1.06E+03	8.93E+02	5.55E+02	4.27E+02
	Std	4.58E+01	2.73E+01	2.46E+01	5.17E+01	8.18E+01	2.38E+01	2.72E+01
F18	Mean	8.78E+02	**5.09E+02**	5.70E+02	2.42E+03	1.76E+03	7.05E+02	5.14E+02
	Std	3.65E+01	3.35E+01	7.48E+01	1.55E+02	2.24E+02	4.05E+01	3.11E+01
F19	Mean	2.33E+03	5.28E+02	5.24E+02	4.05E+05	6.73E+04	5.65E+02	**5.12E+02**
	Std	8.52E+02	2.08E+01	2.76E+01	1.47E+05	6.40E+04	9.14E+01	3.56E+00
F20	Mean	6.15E+02	6.14E+02	6.13E+02	6.15E+02	6.14E+02	6.13E+02	**6.13E+02**
	Std	1.92E-01	6.02E-01	4.53E-01	1.59E-01	5.56E-01	7.55E-01	5.92E-01
F21	Mean	2.78E+03	1.61E+03	1.70E+03	3.18E+03	2.75E+03	1.94E+03	**1.58E+03**
	Std	1.02E+02	2.96E+02	2.95E+02	1.49E+02	1.34E+02	1.23E+02	3.23E+02
F22	Mean	9.03E+03	**4.08E+03**	5.01E+03	8.80E+03	7.64E+03	8.05E+03	4.54E+03
	Std	3.87E+02	6.25E+02	9.69E+02	2.67E+02	6.39E+02	8.31E+02	7.49E+02
F23	Mean	9.21E+03	6.97E+03	7.99E+03	9.22E+03	7.83E+03	8.74E+03	**5.97E+03**
	Std	3.39E+02	8.64E+02	1.17E+03	3.80E+02	6.33E+02	5.81E+02	7.79E+02
F24	Mean	1.30E+03	1.28E+03	**1.26E+03**	1.37E+03	1.31E+03	1.26E+03	1.28E+03
	Std	2.86E+00	8.18E+00	8.13E+00	1.68E+01	1.13E+01	6.03E+00	9.09E+00
F25	Mean	1.42E+03	1.39E+03	**1.38E+03**	1.49E+03	1.42E+03	1.39E+03	1.39E+03
	Std	3.67E+00	7.81E+00	8.67E+00	9.46E+00	7.10E+00	6.57E+00	9.04E+00
F26	Mean	**1.46E+03**	1.54E+03	1.51E+03	1.56E+03	1.57E+03	1.56E+03	1.47E+03
	Std	1.08E+01	6.85E+01	7.07E+01	7.68E+01	6.09E+01	3.17E+01	8.08E+01
F27	Mean	2.65E+03	2.43E+03	**2.25E+03**	3.32E+03	2.67E+03	2.28E+03	2.38E+03
	Std	3.36E+01	8.42E+01	1.17E+02	2.74E+02	8.29E+01	8.03E+01	8.94E+01
F28	Mean	3.90E+03	2.81E+03	2.66E+03	5.39E+03	4.96E+03	2.95E+03	**2.29E+03**
	Std	1.61E+02	6.80E+02	2.40E+02	2.85E+02	3.87E+02	3.36E+02	4.11E+02
Average Rank		5.250	2.893	2.643	6.464	5.25	3.714	**1.786**

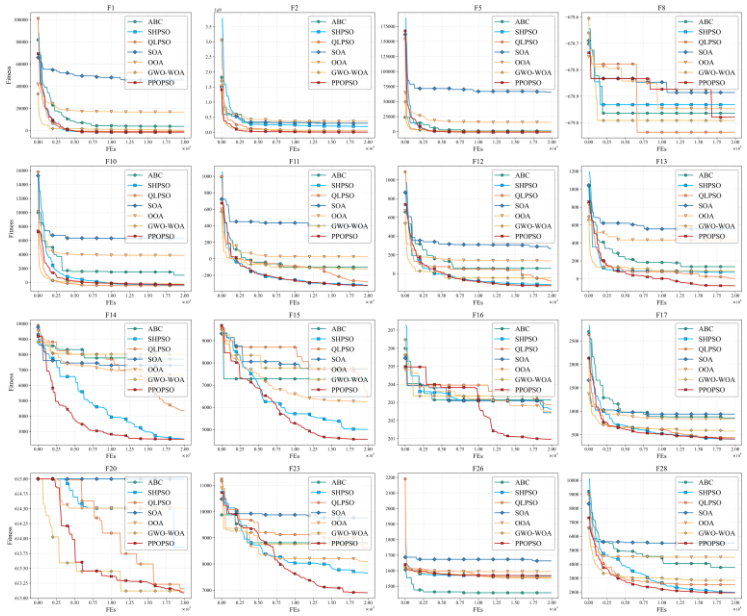

Fig. 5. curves of some functions.

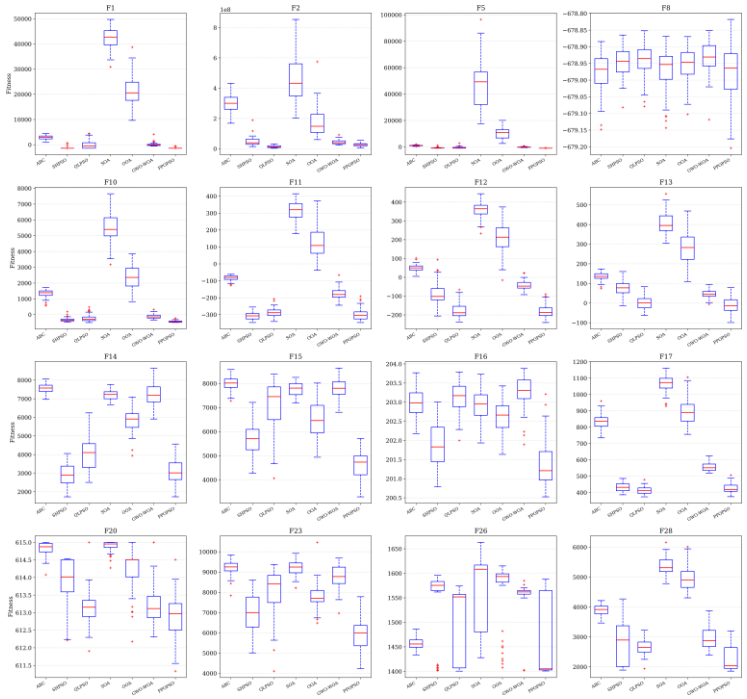

Fig. 6. Boxplots of some functions.

To further validate the performance of our proposed PPOPSO algorithm, we conducted pairwise statistical comparisons against six other well-known algorithms using the sign test[4]. Table 5 summarizes these comparison results. At a significance level of $\alpha = 0.05$, all calculated p-values are far below the significance threshold of 0.05. This indicates that PPOPSO exhibits a significant advantage compared to the other algorithms.

Table 5. Sign Test Results

PPOPSO	ABC	SHPSO	QLPSO	SOA	OOA	GWO-WOA
WIN	27	23	20	28	27	21
LOSS	1	5	8	0	1	7
p-value	2.16e-07	9.12e-04	3.57e-02	7.45e-09	2.16e-07	1.25e-02

5 Conclusion

This paper introduces a Reinforcement Learning (RL)-based dynamic parameter control strategy to enhance the adaptive capabilities and performance of the multi-subgroup cooperative Particle Swarm Optimization (PSO) algorithm. By framing parameter adaptation as a sequential decision-making process, an RL agent was trained to autonomously adjust key behavioral parameters for each PSO subgroup in real-time. This approach replaces manual tuning, enabling each subgroup to dynamically balance exploration and exploitation based on its unique state and historical performance, while an information exchange mechanism facilitates the search for the global optimum. Experimental results demonstrate that this RL-empowered method significantly improves the robustness and solution efficiency of the multi-particle swarm system, particularly on complex optimization tasks. Future work will focus on designing more sophisticated reward functions and state representations, exploring the transfer learning capabilities of the RL agent across different problems, and expanding its control to other architectural aspects of the algorithm, such as subgroup size and migration strategies.

References

1. Bäck, T.H., Kononova, A.V., van Stein, B., Wang, H., Antonov, K.A., Kalkreuth, R.T., de Nobel, J., Vermetten, D., de Winter, R., Ye, F.: Evolutionary algorithms for parameter optimization thirty years later. Evolutionary Computation 31(2), 81–122 (2023)
2. Dehghani, M., Trojovsk`y, P.: Serval optimization algorithm: a new bio-inspired approach for solving optimization problems. Biomimetics 7(4), 204 (2022)
3. Dehghani, M., Trojovsk`y, P.: Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. Frontiers in Mechanical Engineering 8, 1126450 (2023)
4. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1(1), 3–18 (2011)

5. Engelbrecht, A.P.: Heterogeneous particle swarm optimization. In: Swarm Intelligence: 7th International Conference, ANTS 2010, Brussels, Belgium, September 8-10, 2010. Proceedings 7. pp. 191–202. Springer (2010)
6. Karaboga, D., Akay, B.: Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization. In: Innovative production machines and systems virtual conference (2009)
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 international conference on neural networks. vol. 4, pp. 1942–1948. ieee (1995)
8. Li, P., Hao, J., Tang, H., Fu, X., Zhen, Y., Tang, K.: Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms.IEEE Transactions on Evolutionary Computation (2024)
9. Liang, J.J., Qu, B., Suganthan, P.N., Hernández-Díaz, A.G.: Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou,China and Nanyang Technological University, Singapore, Technical Report 201212(34), 281–295 (2013)
10. Lin, S., Liu, A., Wang, J., Kong, X.: An intelligence-based hybrid pso-sa for mobile robot path planning in warehouse. Journal of Computational Science 67, 101938 (2023)
11. Liu, Y., Lu, H., Cheng, S., Shi, Y.: An adaptive online parameter control algorithm for particle swarm optimization based on reinforcement learning. In: 2019 IEEE congress on evolutionary computation (CEC). pp. 815–822. IEEE (2019)
12. Obadina, O.O., Thaha, M.A., Althoefer, K., Shaheed, M.H.: Dynamic characterization of a master–slave robotic manipulator using a hybrid grey wolf–whale optimization algorithm. Journal of Vibration and Control 28(15-16), 1992–2003 (2022)
13. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
14. Shakya, A.K., Pillai, G., Chakrabarty, S.: Reinforcement learning algorithms: A brief survey. Expert Systems with Applications 231, 120495 (2023)
15. Shami, T.M., El-Saleh, A.A., Alswaitti, M., Al-Tashi, Q., Summakieh, M.A., Mirjalili, S.: Particle swarm optimization: A comprehensive survey. Ieee Access 10,10031–10061 (2022)
16. Tung, C.C., Lai, Y.Y., Chen, Y.Z., Lin, C.C., Chen, P.Y.: Optimization of mechanical properties of bio-inspired voronoi structures by genetic algorithm. Journal of Materials Research and Technology 26, 3813–3829 (2023)
17. Wang, S., Liu, G., Gao, M., Cao, S., Guo, A., Wang, J.: Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators. Information Sciences 540, 175–201 (2020)
18. Zhang, W., Xiao, G., Gen, M., Geng, H., Wang, X., Deng, M., Zhang, G.: Enhancing multi-objective evolutionary algorithms with machine learning for scheduling problems: recent advances and survey. Frontiers in Industrial Engineering 2, 1337174 (2024)
19. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical problems in engineering 2015(1), 931256 (2015)