Redesigning the Particle Swarm Optimization Algorithm as a Reinforcement Learning Methodology

Okkes Tolga Altinoz[1][0000−0003−1236−7961]

Ankara University, Ankara 6830, Turkiye taltinoz@ankara.edu.tr

Abstract. Modern optimization algorithms consist of algorithms that have been developed, improved, and expanded for different problems with specific properties as extensive, stochastic, noise multi, many, bi-level, as well as traditional optimization problems. These relatively complex modern algorithms are mainly based on traditional numerical methods. Their complex structures are increasingly being extended with learning-based methods as they can be used in conjunction with machine learning methods, particularly reinforcement learning, and are also used within these methods. These algorithms can be evolutionary or nature-inspired algorithms. Among nature-inspired algorithms, one of the most widely used is the Particle Swarm Optimization (PSO) algorithm. The PSO algorithm has been used in conjunction with machine learning algorithms for solving the problem-oriented applications; and Reinforcement Learning as a machine learning method contains potential to use with PSO. Therefore, in this study, the PSO algorithm will be redesigned as a reinforcement learning framework. The PSO in a different framework will be evaluated on benchmark problems where the innovation suggestions will be made regarding the parameters and formulations of the algorithm according to their areas of use, its performance will be evaluated, and improvement suggestions will be presented.

Keywords: Particle Swarm Optimization · Reinforcement Learning · Learning-based Optimization · Machine Learning · Single-objective · Optimization.

1 Introduction

Teaching–learning-based optimization methods are known for simplicity and fast convergence with exploration-exploitation balance and preferred engineering applications. Optimization algorithms have been evaluating with the machine learning algorithm to form learning-based optimization algorithms. Since learning is essential for these algorithms, modern methods prefer reinforcement Learning framework as the learning part of the proposed methods. In [1], Reinforcement Learning (RL) assisted Evolutionary Algorithms (EA) have reviewed as a survey. In the paper, the usage of RL with the evolutionary algorithm classified with the usage of RL. The classification is established as integration of RL, its strategy, attribute setting and application domains. The classification has made with the direct or indirect integration, continuous or binary application, setting the objective function, reward function or action/policy setting, multi operator selection, multi population selection parameter adaptation are the main themes of these categories. However, all these categories are related to the hybrid usage of the RL or EA with each other in or instead of an operator in RL or EA.

RL and nature-inspired swarm algorithm Artificial Bee Colony (ABC) has proposed and applied to Job-shop scheduling problem [2]. In the paper the aim is to split jobs to multiple subplots so transported separately. The problem has divided into two stages and the second state RL is applied

to best scheme for the subplot, directly. In [3], hybrid flow-shop scheduling problem (DHHFSP) has utilized with Q-learning and Particle Swarm decomposition where the proposed multiobjective framework combines multi directional particle swarm global search with reinforcement learning-based local search.

In [4], tunnel deformation for urban underground safety has been investigated as engineering problem by solving the problem from finite-difference analyses using a Plastic Hardening (PH) soil model. In the paper another hybrid methodology/framework has proposed by the authors so that a hybrid machine-learning framework integrated with particle swarm optimization (PSO), convolutional neural networks (CNN), and extreme gradient boosting (XGBoost). Another application for the RL improved methods with Tabu search has presented for coloring problem in [5] where it was tested on 60 benchmark instances. In [6], an electrical problem for the shipboard has proposed and modeled this problem as shipboard multi-angle photovoltaic system with the synergistic Hybrid Evolutionary Algorithm.

A hybrid framework that integrates Quantum-Inspired Particle Swarm Optimization (QIPSO) with Deep Q-Network (DQN) reinforcement learning has been proposed and its performance has evaluated on five real-world scientific workflows, Montage, SIPHT, CyberShake, Epigenomics, and Inspiral across heterogeneous edge nodes [7]. In [8], time series prediction for six have solved with a novel proposed algorithm named as Hybrid Optimization Expert System that integrates multiple evolutionary algorithms (Arctic Puffin Optimization (APO) algorithm, Bloodsucking leech optimization algorithm (BSLO), Enterprise Development Optimization Algorithm (EDO), The Flying Lizard Optimization (FLO)) and employs BiLSTM (SJ-LSTM)a transmission mechanism, memory system, and punishment system to achieve collaborative optimization. In [9], like other studies proposes a novel multi-exemplar driven teaching learning based optimization has proposed and applied to benchmark problems CEC2014, CEC2019, and real word application benchmark problem set. A reinforcement learning-based a risk aware portfolio optimization method named Risk aware Trader has proposed as a crucial decision-making component of intelligent financial system [10]. The performance of the proposed algorithm demonstrated on numerical experiments with some public datasets, namely the Dow Jones Industrial Average and the China Securities Index.

The Q-learning method with Reinforcement learning contains potentials to be used with PSO algorithm. Unlike the previous studies it is possible to remodeled the PSO working with Reinforcement learning to select the best possible parameter sets among the action set based on diversity of the position of the population on the decision space and the improvement of the objective function on the objective space. For this reason in this research Reinforcement Learning-based Particle Swarm Optimization is proposed for single objective problems.

This paper is organized as five sections beginning with the introduction and conclusion as the final section. Section two is given for Problem statement that Reinforcement Learning, and Particle Swarm Optimization algorithms are briefly explained. In Section 3 PSO is redesigned to be used with Reinforcement Learning and in Section 4 the implementation results for this research is presented.

2 Problem Statement

RL, one of the machine learning methods, has been shown to be effectively used in conjunction with optimization algorithms in light of studies in the literature. These uses ensure that optimization problems are solved more efficiently and/or more accurately. However, in the literature, RL optimization algorithms are used in conjunction with other methods. RL is either used in place of one or more operators or to reduce the performance and computational cost of the optimization

algorithm. In this study, the PSO algorithm, one of the most well-known algorithms among swarm intelligence methods, will be converted into an RL framework. The algorithm will be transformed into an RL-like form. Subsequently, an assessment will be made of which section of the literature corresponds to this framework. In this respect, part of the research presented in this study offers a different perspective on the literature review.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a learning methodology (other machine learning methods are unsupervised learning and supervised learning) that maps the states from the environment to the actions transfer to the environment by using an assessment function called the reward. In RL the method called agent has two common input and one output. The inputs are the observations from the environment and the rewards, the output is the action. The agent contains two main operators called the policy and RL algorithm. The function takes the observations from the environment and decide an action, that function called policy. The RL algorithm is a methodology to update the policy based on the observation generated action and the reward function. Fig. 1 presents the graphical representation of the RL.

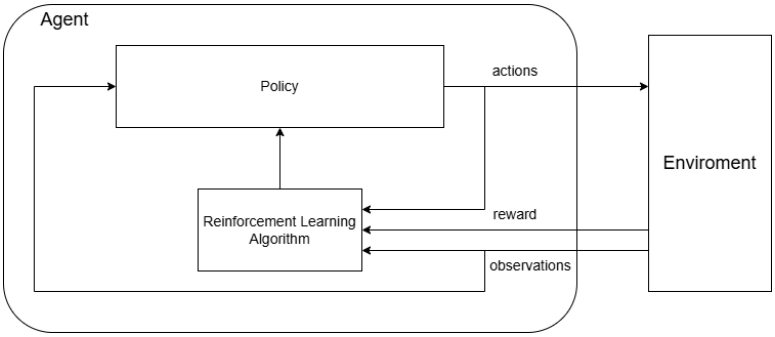

Fig. 1: General Graphical Description for Reinforcement Learning Framework.

As given in Fig. 1, the aim of the policy is to generate a n action for the desired operator by using the given observation and updates from the RL algorithm. The RL algorithm gets the action generated from policy, the reward function and the observations to update the policy to get the desired operator.

The RL methodology contains four works. These are environments, reward, policy, and training. The environment is the remaining part of the overall system except RL. This is the part that RL submits the action and gets the observations and reward. However, based on the evaluation of the environment the RL gets model-free RL and Model-based RL methods. In model free RL, the agent does not know the environment (for example dynamics of the environment) only gets the observations and policy. That means the agent do not know prior information related to the environment, therefore the agent must search the environment deeply to get a better reward. In model-based RL the agent knows (maybe partly) some parts or everything related to the environment. For example, a total map for an apart of the map for the path planning problem. Also, the environment is considered in a simulation environment of the real-life environment. The real environments present the

accurate performance form the RL however the damage to the environment is inevitable in meany engineering applications. In simulation environment is a low-cost RL application however it is not every time possible to get an accurate solution when transferred to the real word.

The reward is the numeric version of the evaluation of the action generated by the policy. It is the degree of the evaluations of how good the performance of the environment for a given action. The policy could be sparse, discrete, or continuous functions. An important part related to the selection of the reward function is to get a tradeoff between exploration and exploitation behavior.

The policy is a function that maps the observations to the actions. The policy could be direct or indirect that in direct manner the observations directly evaluated inside the policy and the indirect manner the observations are applied to the metrics and metric values are inserted to the policy. If the actions are discrete or represented by discrete states it is possible o define a look-up table. This look-up table is called Q-table that maps states/observations to actions. Instead of continuous function and Q-table, neural networks can be used as policy that is the most preferred approach in AI applications.

The policy structured into three different approaches. These are policy function based, value function based and actor-critic approaches. The policy function-based learning is the entire learning method for a single structure is the entire policy like neural network. For example, the output of the neural network are the discrete events on a system. Similarly, policy gradient methods that gradient of the rewards and/or the policy uses to decide the direction of the better reward value. Value function-based policy is a system that not generate action solely. Instead, it generated some value and based on that value the action is decided. for example, the Q-table and the bellman equation to update the entities of the Q-table can be given.

Actor-critic policy and learning is a continuous network so that the actor generates the action and the critic predicts the reward value. In general, the neural network is the actor because it directly generated the actions. As critic neural network also utilize to predict the reward value. Similarly, both networks trained with the observations.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was initially proposed by Kenndy and Eberhart in 1995 [11]. The main motivation of the algorithm is to integrate the nature inspired swarm intelligence to the search algorithm that formed an optimization algorithm. The algorithm is depended on the movement of the particles on the objective space based on the movements of the members in an animal flock or swarm. The change of positions of the members in a flock becomes solution candidates on the population-based PSO algorithm. Since the PSO is relatively simple to implement and has relatively low number of parameters that needed to adjusted to get a desired performance; are two reasons to get attention from the engineering problems as a stochastic optimization methodology. The PSO algorithm contains position and velocity update rules as given in Eq. 1.

$$\begin{aligned} v(k+1) &= wv(k) + c_1 r_1 \left(pb - x(k)\right) + c_2 r_2 \left(gb - x(k)\right) \\ x(k+1) &= x(k) + v(k+1) \end{aligned} \tag{1}$$

where v is the velocity, x is the position, k is the iteration index, pb is the personal best from the beginning of the iteration to the kth iteration, w, c_1 and c_2 are the control parameters with two random number generators r_1 and r_2 within $[0, 1]$.

The performance of the PSO algorithm have been improving through more than thirty years after the proposing the PSO in terms of exploration and exploitation. Algorithm 1 gives the pseudo-code for the PSO algorithm. Since PSO is a population-based algorithm, it begins with the initial definition of the parameters and position. Next the algorithm begins with global and personal best member detection and velocity update and position update rules. This procedure continues until the maximum number of iterations is reached. This algorithm suffers from the local search with local optimum, early convergence and lower accuracy. Therefore, many variants have proposed to improve the performance. The impact of some of the main improvements will be discussed after the redesigning of PSO in the next sub-section.

Algorithm 1 Particle Swarm Optimization (PSO)

Select random number generator, number of members in population N, maximum number of iterations M and parameters w, c_1 and c_2
for particle i **do**
  Define initial position for each particle $x(0)$
  Calculate Objective function $f(x(0))$
end for
for $k = 1$ increase k $k = k + 1$ **do**
  Update Personal best pb
  Update Global best gb
  Update Velocity

$$v(k+1) = wv(k) + c_1 r_1 \left(pb - x(k)\right) + c_2 r_2 \left(gb - x(k)\right) \tag{2}$$

  Update Position

$$x(k+1) = x(k) + v(k+1) \tag{3}$$

  Calculate Objective function $f(x(k))$
end for

3 Redesigning Particle Swarm Optimization Algorithm

The Algorithm 1 given for the PSO presents the flow of the algorithm in a sequential manner. The PSO algorithm mainly contains the update rule with the parameters the objective function. The nature of this algorithm suitable to map the sequential algorithm to the RL framework. Fig. 2 graphically demonstrate the PSO algorithm in the form of RL framework.

As given in the figure, RL framework has two main blocks that are policy and the learning algorithm with reward function. It is clear to state that the reward function easily maps to the reward of the optimization problem. The action is defined as the velocity and the velocity update rule is the policy of this structure. Therefore, the position is the observation from the environment. The position also the solution candidate that is evaluated from the objective function also called the reward function.

The learning algorithm has four parameters that have to be set. Luckily, conventional PSO has constant values for c_1, and c_2. The personal best value and the global best value calculated from the population by using the min function by using the reward function. The other parameter w is the weighting multiplier set to be decreasing with the iteration.

In the following two sub-sections the proposed improvements for the PSO are evaluated with respect to the policy and learning algorithms. Therefore, in this research the improvements on the

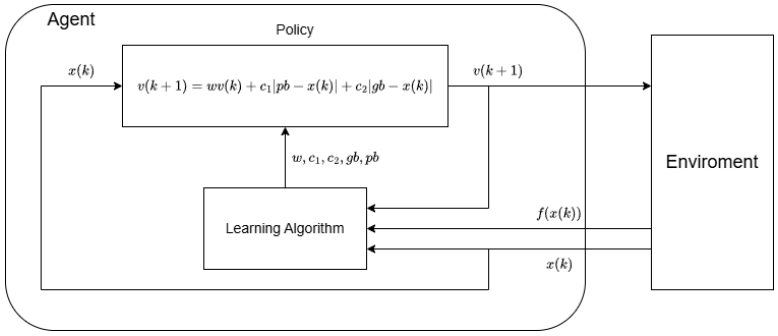

Fig. 2: General Graphical Description for the Particle Swarm Optimization Algorithm formed as Reinforcement Learning Framework.

algorithm categorized into two sets and it is possible to categorized other variants of the PSO. However, to maintain the content of the research only the well-known methods are selected and discussed.

3.1 Learning Algorithms

The learning algorithms are sub-routines that adjust the parameters of the PSO algorithm. Therefore, the learning is to set these values based on the observation actions and reward function. As conventional way, the learning is the decision method for the parameters related to the policy of the PSO variant. In general, the intelligence methods to decide these parameters corresponding to the learning algorithm. In general, three main groups are well known which are parameter update, inertia weight and constriction factor. There are other methods based on policy of the algorithm.

Parameter Update: Clerc's study [12] has the possible selection of the parameters c_1 and c_2 so that it is possible to considered the corresponding values of these two parameters gives the decision between cognitive-dominant ($c_1 > c_2$); social-dominant ($c_1 < c_2$) and balanced behavior ($c_1 = c_2$) (in [12] it was stated that $c_1 = c_2 = 2.05$) between personal best and global best tendency between particles [13].There are many possible selection for these values like in [14] $c_1 = c_2 = 1.49$ parameter set was selected.

Inertia Weight: In [15], a weighting parameter is defined for the PSO so that as given in Eq.1 the weight multiplied with the previous velocity. This parameter uses to balance exploration and exploitation. The study of [16] the suggestion for the weight is in the range between 0.5 and 0.8. The adaptive exponential linear and nonlinear variations of the weight based on iteration or rank can be selected as a weighting variant. In [13] these methods have given as a summarized table, and also given in [17].

Constriction Factor: The parameter is given to multiply the overall velocity update rule (without the weight) to increase convergence and stability of the PSO algorithm suggested in [12]. The common value for this parameter is 0.729 [12].

3.2 Policy

After redesigning the PSO as discussed in the previous section, the PSO algorithm have divided into two parts, the policy and the learning. In this subsection the policy will be discussed. The policy

is the function that generates the action therefore for the PSO as explained it is corresponding to the velocity update formulations for each variant.

Velocity Clamp: The first and easy part is the restrict the obtained velocity. This action corresponds to the actor-critic manner so that the critic related to the generated action is made by comparing the maximum and minimum velocity values. The restriction to the velocity is a common methodology to restrain the position of the population. The maximum and minimum velocity helps the algorithm to remains in a given position range [18]. It is possible to change this maximum and minimum values at each iteration.

Velocity Update Rules: The velocity update rules changed, improved, merged with other operator to get more accurate solution with a smaller number of iterations. As an example in [19] two mutation operators merged with PSO to update velocity. Similarly in [20], differential mutation operator with social learning mechanism utilized with PSO. In [21], the velocity update rule is changed by introducing some novel parameters like in [22] where a third summation is added to the velocity update rule with a new parameter c_3 and a direction value.

3.3 Proposed RL-PSO Algorithm

The proposed RL-PSO algorithm begins with the random initialization of the positions and velocity. Then the observations are determined. Observation space is selected as six states. The state is determined from the global best position $s(t) = f(g^{best}(t) - g^{best}(t-1))$. Next the action space is decided. The action space contains three parameters w, c_1 and c_2. These parameters took the values inside $[0.1, 2]$ with 10 discrete and equally divided values. Next based on the selected parameter the position and the velocity of the members are updated.

$$v_{i,d}(t+1) = w \cdot v_{i,d}(t) + c_1 r_1 \big(p_{i,d}^{best} - x_{i,d}(t)\big) + c_2 r_2 \big(g_d^{best} - x_{i,d}(t)\big), \tag{4}$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1), \tag{5}$$

Unlike previous discussion, in this RL-PSO the reward function is defined as $r(t) = g^{best}(t-1) - g^{best}(t)$ which means the reward is the difference between global best objective function values at the current and the previous iteration. Next, the Q-table is updated and new state is determined. In RL-PSO algorithm the action parameters are determined from Q-learning agent.

The update of the Q-table is made by Bellman function like the conventions Q-function training methodology given as

$$Q(s(t), a(t)) \leftarrow Q(s(t), a(t)) + \alpha \Big[r(t) + \gamma \max_{a' \in A} Q(s(t+1), a') - Q(s(t), a(t)) \Big], \tag{6}$$

where $\alpha \in (0, 1)$ is the learning rate, and $\gamma \in (0, 1)$ is the discount factor. The policy selection is made from ε-greedy policy.

Unlike conventional way that the velocity update rule is the policy; in RL method the policy is the ε-greedy and the training rule is given in Eq. 6. The next section shows the implementation of RL-PSO algorithm and comparison between PSO algorithm is proposed.

In this framework, the difference between global best objective value of the current iteration and the previous iteration is selected as the improvement. In addition the average of the standard deviation of the members in the population on the decision space is the diversity of the population. Based on improvement and diversity of the solutions the states are selected. The Q-table contains

6 states and 8000 actions. the actions are formed from three parameters w, c_1 and c_2 and their 20 different values. The Q-table values are updated with Bellman functions and based on the current state and the Q values the action is selected among the 8000 actions.

Table 1: State Transition Logic Based on Improvement and Diversity

State	**Improvement**	**Diversity**	**Interpretation**
1	$> 10^{-2}$	> 1	Fast Growth
2	$> 10^{-4}$	> 1	Approximate Steady State
3	> 0	≤ 1	Steady state
4	≤ 0	> 1	Wide Stagnation
5	≤ 0	> 0.1	Narrow Stagnation
6	Else	≤ 0.1	End

Table 1 gives the state ids and their approximate interpretations with respect to the ranges of the improvement and diversity. As the improvements are increased with the diversity of the populations that means searching a relatively large area of the objective space (exploration) and as the improvement reduces, became smaller with a more crowded populations means the solution becomes converging therefore the algorithm is approximately in the steady state.

4 Implementation

In this section, the brief information related to the benchmark problems and their properties will be explained. Next, the implementation results and their convergence will be reported as the results and discussions.

4.1 Benchmark Problems

In this research to demonstrate the performance of the proposed RL-PSO algorithm explained in the previous section is applied to the CEC 2017 benchmark problem set defined in [23]. The benchmark set contains 30 problems however problem 2 is deleted after the definition of the problems (in total there are 29 problems) where the codes for the Matlab file can be found in [24]. The search space for all these benchmark problems are in $[-100, 100]^D$ where D is the dimension of the search space which is equal to 30 for all problems. The problems are categorized into four groups, problem 1 and 2 are unimodal Functions, Problem 3-9 are simple multimodal functions, problems 10-19 are hybrid functions and problems 20 - 29 are composition functions. First 9 problems are actually the shifted and rotated of the conventional benchmark problems like Rosenbrock and Rastrigin's Functions. Problem 10-19 are hybrid problems so that the linear combination of the different basic functions is weighted and summed as sub components of these benchmark problems. Finally, there are composition functions where these functions are like hybrid functions with weights, biases and shifted functions. Unlike hybrid problems, in composition problems all basic functions have been used as shifted and rotated. In [23], the mathematical expressions and their search spaces are graphically demonstrated. The optimum values for each benchmark problems are equal to problem ID multiply by 100. For example, the optimum solution for the final problem is 2900.

4.2 Results and Discussions

In this research Reinforcement Learning is used to determine the best w, c_1 and c_2 parameters. The obtained algorithm named as RL-PSO algorithm. The performance of this algorithm will be compared with PSO meta-algorithm under same conditions. All algorithms have 100 members in their population 30 and iteration is 200. The implementations are repeated 30 times and the statistics of these results are reported numerically. In addition, Wilcox statistical test is provided to show the significant of the results. for the PSO algorithm the parameters are selected as $w = 0.7$, $c_1 = 2$, and $c_2 = 2$. The implementations reevaluated and Table 2 gives the results numerically and Fig 4 shows the average convergence plots for all of the benchmark problems.

Table 2 gives the statistical properties of the 30 independent run with mean and standard deviation. In addition, Wilcoxon significant test results are added and it is numerically showed that the proposed algorithm is better than PSO for 12 problems and almost same for 11 other problems. Also, for the rest of the problems PSO gives better objective values, performance. For unimodal and simple multimodal problems, the proposed method presents better results however for composition and hybrid problems the performance of PSO and RL-PSO gives almost same performance.

Figure 4 gives the selected actions for each problems. Each plot in Figure 4 gives for each benchmark problems. The plots are given for each iteration that means for each iteration a different action is selected based on the Q-value. These plots are among the 30 independent values the most frequently selected action for a given iteration. As a general perspective, it can be observed from the figure that as the iteration increased in number the actions becomes more converged value for a better performance cases.

5 Conclusion

In this research the PSO algorithm parameters w c_1 and c_2 determined with the aid from reinforcement learning so that the Q-table is generated. From the table the best parameters are obtained for PSO algorithm under CEC 2017 test problems. The results are numerically and graphically compared with each other. The results showed that RL-PSO presents mostly better results. However, since PSO is relatively simple nature and easy to implement and has lower computational necessities, it is possible to comment that even RL-PSO gives better results the desired computational resources is relatively highly for this performance. As future study, learning-based algorithms are more deeply discussed to get a better single and multi-objective problems.

References

1. Yanjie Song, Yutong Wu, Yangyang Guo, Ran Yan, Ponnuthurai Nagaratnam Suganthan, Yue Zhang, Witold Pedrycz, Swagatam Das, Rammohan Mallipeddi, Oladayo Solomon Ajani, Qiang Feng: Reinforcement learning-assisted evolutionary algorithm: A survey and research opportunities, Swarm and Evolutionary Computation, **86**(2024), 101517, (2024), https://doi.org/10.1016/j.swevo.2024.101517.
2. Yibing Li, Cheng Liao, Lei Wang, Yu Xiao, Yan Cao, Shunsheng Guo: A reinforcement learning-artificial bee colony algorithm for flexible job-shop scheduling problem with lot streaming, Applied Soft Computing, **146**(2023), 110658, (2023) https://doi.org/10.1016/j.asoc.2023.110658.
3. Wenqiang Zhang, Sen Yang, Mingzhe Li, Yashuang Mu, Guohui Zhang, Mitsuo Gen, Peng Li: A hybrid multiobjective particle swarm optimization with Q-learning-driven local search for distributed heterogeneous hybrid flow-shop scheduling problem with worker–machine–environment collaboration. Journal of Manufacturing Systems,**85**(2026), 1–21, (2026) https://doi.org/10.1016/j.jmsy.2025.12.024.

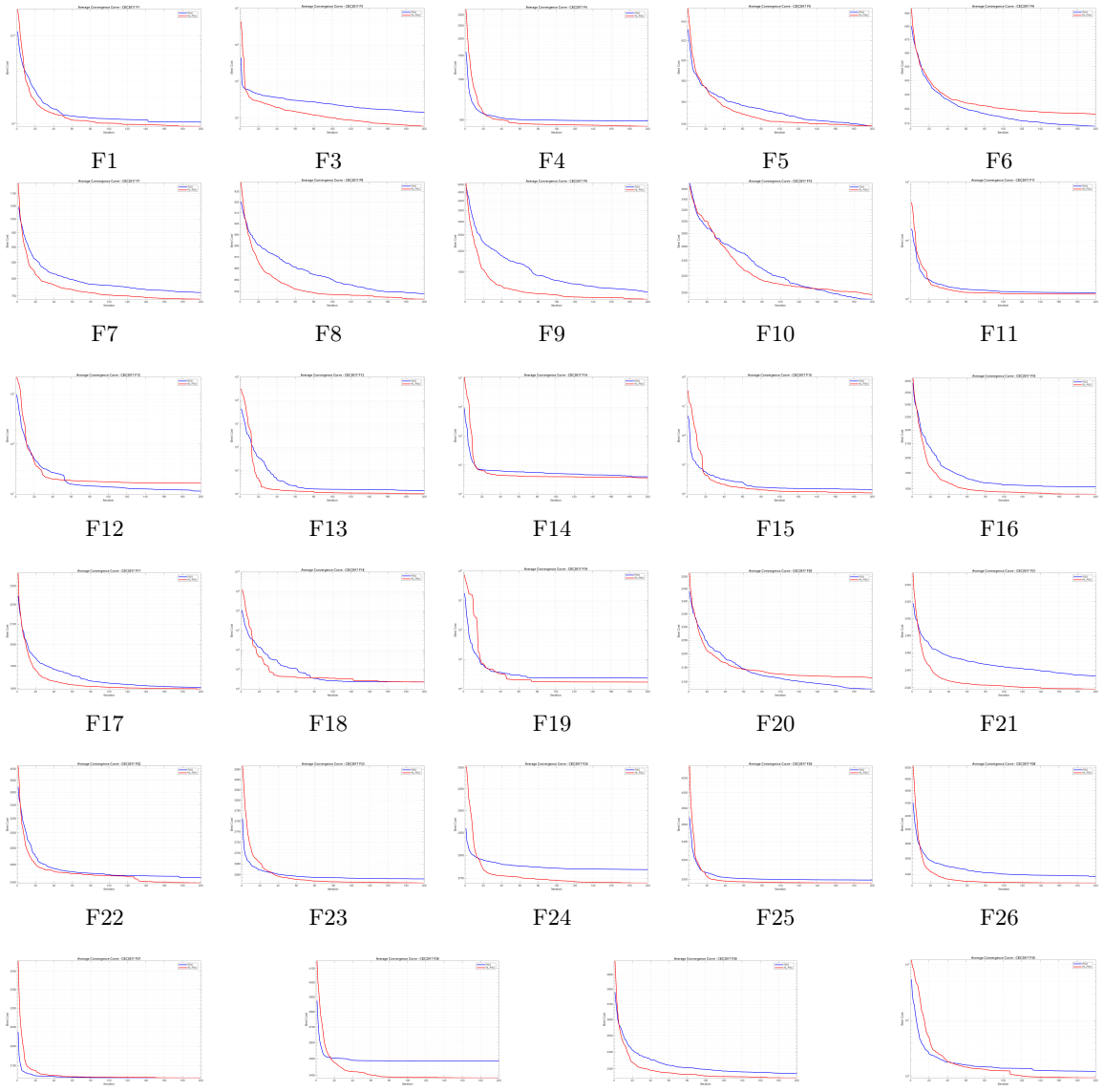

Fig. 3: Average convergence curves for 29 CEC 2017 functions

Table 2: Statistical properties of the optimization algorithms based on their mean and standard deviation of the objective values.

Prob. ID	PSO	RL-PSO
1	01.37e+09 (01.21e+09)	03.55e+08 (06.63e+08) $-$
3	13.78e+02 (95.31e+02)	50.08e+02 (70.83e+02) $-$
4	49.05e+01 (11.89e+01)	44.03e+01 (54.74e+00) $-$
5	53.83e+01 (10.594e+00)	54.20e+01 (18.81e+00) $+$
6	60.88e+01 (06.86e+00)	61.25e+01 (10.34e+00) $+$
7	75.89e+01 (14.51e+00)	75.64e+01 (42.93e+00) $-$
8	83.75e+01 (10.67e+00)	83.86e+01 (18.83e+00) $\approx$
9	10.45e+02 (23.14e+01)	12.12e+02 (45.81e+01) $+$
10	19.32e+02 (30.05e+01)	19.45e+02 (39.62e+01) $\approx$
11	13.29e+02 (27.27e+01)	13.06e+02 (55.25e+01) $\approx$
12	01.77e+07 (05.35e+07)	01.35e+06 (02.69e+06) $-$
13	14.56e+03 (14.22e+03)	93.24e+02 (10.09e+03) $-$
14	38.10e+02 (50.81e+02)	55.97e+02 (71.67e+02) $+$
15	13.00e+03 (19.94e+03)	10.67e+03 (11.97e+03) $-$
16	17.82e+02 (12.55e+01)	18.16e+02 (16.39e+01) $\approx$
17	18.10e+02 (61.01e+00)	17.95e+02 (62.33e+00) $\approx$
18	30.14e+03 (13.89e+03)	42.39e+02 (74.82e+03) $+$
19	33.74e+03 (56.59e+03)	15.10e+02 (21.76e+03) $-$
20	20.84e+02 (54.72e+00)	21.37e+02 (67.32e+00) $+$
21	23.39e+02 (09.55e+00)	23.29e+02 (37.83e+00) $\approx$
22	23.75e+02 (19.73e+01)	23.58e+02 (14.98e+01) $-$
23	26.50e+02 (23.15e+00)	26.51e+02 (27.38e+00) $\approx$
24	27.72e+02 (42.35e+00)	27.73e+02 (61.54e+00) $\approx$
25	29.84e+02 (60.02e+00)	29.81e+02 (97.89e+00) $\approx$
26	33.22e+02 (44.45e+01)	31.45e+02 (34.58e+01) $-$
27	31.07e+02 (11.31e+00)	31.25e+02 (34.83e+00) $\approx$
28	34.44e+02 (14.86e+01)	33.54e+02 (12.61e+01) $-$
29	32.87e+02 (73.23e+00)	32.74e+02 (81.30e+00) $\approx$
30	09.34e+05 (06.78e+05)	05.08e+05 (05.72e+05) $-$
$+/-/\approx$		6/12/11

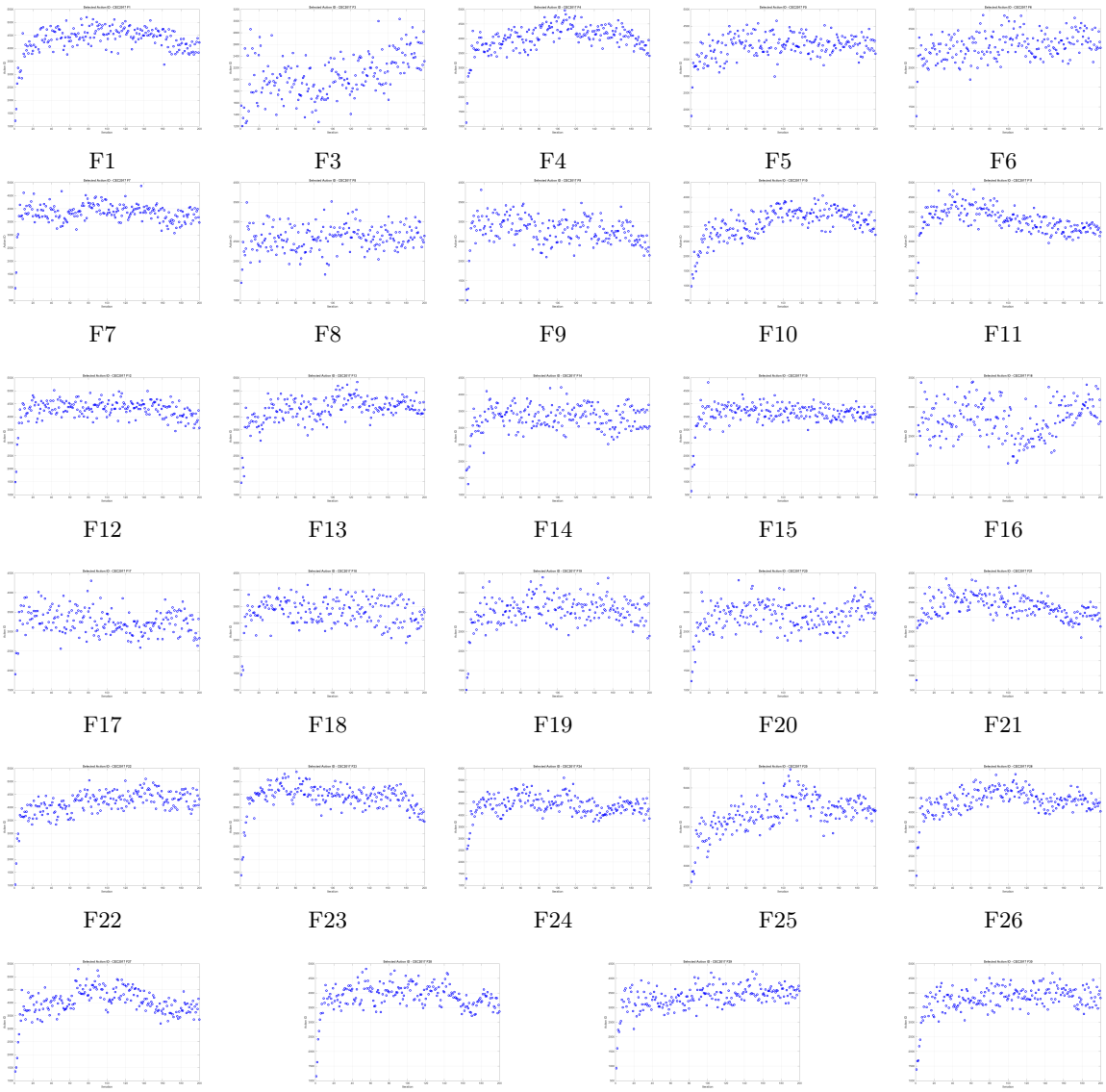

Fig. 4: Average selected Action ID for for 29 CEC 2017 functions

4. Siyu Yin, Zheng Yang, Kunpeng Cao, Ben Wu, Siau Chen Chian: Plastic-hardening constitutive model-based hybrid machine learning framework for three-dimensional tunnel deformation prediction, Tunnelling and Underground Space Technology. **171**(2026), 107415, (2026) https://doi.org/10.1016/j.tust.2025.107415.
5. Zhe Sun, Una Benlic, Mingjie Li, Qinghua Wu: Reinforcement learning based tabu search for the minimum load coloring problem, Computers and Operations Research, **143**(2022), 105745, (2022) https://doi.org/10.1016/j.cor.2022.105745.
6. Qilin Xiang, Lijie Xu, Chengqing Yuan,: A reinforcement learning-based Synergistic Hybrid Evolutionary Algorithm for multi-angle shipboard photovoltaic system MPPT under dynamic navigational shading. Renewable Energy,**260**(2026), 125193, https://doi.org/10.1016/j.renene.2026.125193.
7. Megha Sharma, Abhinav Tomar: QERLO An intelligent Quantum-Enhanced Reinforcement Learning framework for task Offloading in IoT networks. Advanced Engineering Informatics, **70**(2026), 104096, https://doi.org/10.1016/j.aei.2025.104096.
8. Wei, P., Fan, C., Yang, X. et al. HOES: an efficient multi-evolutionary expert system for deep learning model optimization in time series prediction. Sci Rep 16, 527 (2026). https://doi.org/10.1038/s41598-025-30014-4
9. Sharma, A., Lim, W.H., Berghout, T. et al. Process Innovation via Adaptive Multi-exemplar Driven Optimization: Enhancing Engineering Design and Global Optimization Performance. Int J Comput Intell Syst 19, 22 (2026). https://doi.org/10.1007/s44196-025-01086-x
10. Yang, M., Wang, J. and Hu, Y. RiskawareTrader A Reinforcement Learning based Portfolio Optimization for Risk Averter. Int J Comput Intell Syst 19, 25 (2026). https://doi.org/10.1007/s44196-025-01094-x
11. J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.
12. M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, Feb. 2002, doi: 10.1109/4235.985692.
13. Rashmi Sharma, Jagjit Singh Matharu, Kulwinder Singh Parmar: A survey on Particle Swarm Optimization: Evolution, adaptations and practical implementations, Applied Soft Computing, Volume 186, Part A, 2026, 114016, https://doi.org/10.1016/j.asoc.2025.114016.
14. A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," in IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 240-255, June 2004, doi: 10.1109/TEVC.2004.826071.
15. Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA, 1998, pp. 69-73, doi: 10.1109/ICEC.1998.699146.
16. Lin Xueyan and Xu Zheng, "Swarm size and inertia weight selection of Particle Swarm Optimizer in system identification," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Harbin, 2015, pp. 1554-1556, doi: 10.1109/ICCSNT.2015.7491026.
17. T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," in IEEE Access, vol. 10, pp. 10031-10061, 2022, doi: 10.1109/ACCESS.2022.3142859.
18. Russ Eberhart, Pat Simpson, and Roy Dobbins. 1996. Computational intelligence PC tools. Academic Press Professional, Inc., USA.
19. Yonggang Chen, Lixiang Li, Haipeng Peng, Jinghua Xiao, Yixian Yang, Yuhui Shi: Particle swarm optimizer with two differential mutation, Applied Soft Computing, Volume 61, 2017, 314-330, https://doi.org/10.1016/j.asoc.2017.07.020.
20. Xinming Zhang, Xia Wang, Qiang Kang, Jinfeng Cheng: Differential mutation and novel social learning particle swarm optimization algorithm, Information Sciences, Volume 480, 2019, 109-129, https://doi.org/10.1016/j.ins.2018.12.030.

21. Davoud Sedighizadeh, Ellips Masehian, Mostafa Sedighizadeh, Hossein Akbaripour: GEPSO: A new generalized particle swarm optimization algorithm, Mathematics and Computers in Simulation, Volume 179, 2021, 194-212, https://doi.org/10.1016/j.matcom.2020.08.013.
22. Abdullah, M.N. and Bakar, Ab Halim Abu and Rahim, N.A. and Mokhlis, Hazlie and Illias, Hazlee Azil and Jamian, J.J. (2014) Modified Particle Swarm Optimization with Time Varying Acceleration Coefficients for Economic Load Dispatch with Generator Constraints. Journal of Electrical Engineering Technology, 9 (1). pp. 15-26. ISSN 1975-010
23. N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang and B. Y. Qu, "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization" Technical Report, Nanyang Technological University, Singapore And Jordan University of Science and Technology, Jordan And Zhengzhou University, Zhengzhou China, 2016.
24. P-N-Suganthan, CEC2017-BounContrained, Available in https://github.com/P-N-Suganthan/CEC2017-BoundContrained