# RLEPSO:Reinforcement learning based Ensemble particle swarm optimizer*

SHIYUAN YIN, Institute of Semiconductors,CAS, China
YI LIU, Institute of Semiconductors,CAS, China
GUOLIANG GONG†, Institute of Semiconductors,CAS, China
HUAXIANG LU, Institute of Semiconductors,CAS University of Chinese Academy of Sciences China Academy of Sciences brain science and intelligent technology excellence innovation center Beijing Key Laboratory of semiconductor neural network intelligent sensing and computing technology, China
WENCHANG LI, Institute of Semiconductors,CAS, China

Evolution is the driving force behind the evolution of biological intelligence. Learning is the driving force behind human civilization. The combination of evolution and learning can form an entire natural world. Now, reinforcement learning has shown significant effects in many places. However, Currently, researchers in the field of optimization algorithms mainly focus on evolution strategies. And there is very little research on learning. Inspired by these ideas, this paper proposes a new particle swarm optimization algorithm Reinforcement learning based Ensemble particle swarm optimizer (RLEPSO) that combines reinforcement learning. The algorithm uses reinforcement learning for pre-training in the design phase to automatically find a more effective combination of parameters for the algorithm to run better and Complete optimization tasks faster. Besides, this algorithm integrates two robust particle swarm variants. And it sets the weight parameters for different algorithms to better adapt to the solution requirements of a variety of different optimization problems, which significantly improves the robustness of the algorithm. RLEPSO makes a certain number of sub-swarms to increase the probability of finding the global optimum and increasing the diversity of particle swarms. This proposed RLEPSO is evaluated on an optimization test functions benchmark set (CEC2013) with 28 functions and compared with other eight particle swarm optimization variants, including three state-of-the-art optimization algorithms. The results show that RLEPSO has better performance and outperforms all compared algorithms.

CCS Concepts: • **Theory of computation → Reinforcement learning**; **Optimization with randomized search heuristics**.

Additional Key Words and Phrases: Particle Swarm Optimization,Reinforcement Learning,Policy Gradient.

**ACM Reference Format:**
Shiyuan Yin, Yi Liu, GuoLiang Gong, Huaxiang Lu, and Wenchang Li. 2021. RLEPSO:Reinforcement learning based Ensemble particle swarm optimizer. 1, 1 (June 2021), 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*This work was supported by the National Natural Science Foundation of China (U19A2080, U1936106); High-tech project (31513070501, 1916312ZD00902201); CAS Strategic Leading Science and Technology Project (XDA27040303, XDA18040400, XDB44000000);
†corresponding author

Authors' addresses: Shiyuan Yin, yinshiyuan@semi.ac.cn, Institute of Semiconductors,CAS, Beijing, China; Yi Liu, liuyi@semi.ac.cn, Institute of Semiconductors,CAS, Beijing, China; GuoLiang Gong, gongmianjie@semi.ac.cn, Institute of Semiconductors,CAS, Beijing, China; Huaxiang Lu, luhx@semi.ac.cn, Institute of Semiconductors,CAS, University of Chinese Academy of Sciences, China Academy of Sciences brain science and intelligent technology excellence innovation center, Beijing Key Laboratory of semiconductor neural network intelligent sensing and computing technology, Beijing, China; Wenchang Li, liwc@semi.ac.cn, Institute of Semiconductors,CAS, Beijing, China.

## 1 INTRODUCTION

Optimization problems can be found in all areas of science and engineering. As many real-world optimization problems become increasingly complex, it is generally impossible to obtain the optimal solution by exhaustive attack method. So heuristic algorithms are generally used to obtain a good solution. At present, heuristic algorithms are developing rapidly, and many excellent algorithms have emerged, such as artificial bee colony algorithm (ABC) [7], evolutionary strategy (ES) [21], differential evolution (DE) [4], evolutionary programming (EP) [27], genetic algorithm (GA) [18], ant colony optimization (ACO) [2] algorithm and particle swarm optimization (PSO) etc. The performance of these algorithms had been tested on real-parameter optimization benchmark problems [1, 9–11, 13, 17, 25].

Among these algorithms, the particle swarm algorithm has received widespread attention due to its simple parameter configuration and fast convergence speed. Particle swarm optimization algorithm is currently widely used, and its applications include machine learning of neuro-fuzzy system [24], scheduling problems [6, 19], autonomous navigation [5], system identification and control [15, 26], design optimization [8] and so on.

However, the particle swarm optimization algorithm also has many shortcomings, such as premature maturity and prolonged later search process. Therefore, many researchers have made many improvements to the algorithm because of these shortcomings of the particle swarm algorithm.

In fitness-distance-ratio based particle swarm optimizer (FDR-PSO), particles move to nearby particles with higher adaptability, instead of just moving to the global best position [20].

In comprehensive learning particle swarm optimizer (CLPSO), every particle use all the other particles' historical best information to update their own velocity [12].

In Orthogonal learning particle swarm optimization(OLPSO), the exemplar of every particle is generated using orthogonal learning strategy and the best experience of the swarm [28].

In Locally Informed Particle Swarm optimization(LIPS),particles follow local bests(the best experiences of the neighboring particles),instead of the best experience in the swarm, to find the optimum over the search space [22].

In self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC), Every particle use cognitive and social parts to update their velocity.And when they are stagnated in the search space,they will be reinitialized [23].

In heterogeneous particle swarm optimization(HPSO),there is a pool of different search behaviors.Each particle update their velocity using a behavior from the pool randomly [3].

In Ensemble particle swarm optimizer [16], there are five PSO variants integrated. The particles in the algorithm will be divided into two groups. The bigger group will update themselves using a PSO variant according to their success rate. And the smaller group will use CLPSO to update their particle.

Although these algorithms have improved the particle swarm's performance in many ways because humans set their running parameters, these algorithms still have redundancy. At present, reinforcement learning is developing rapidly, and better strategies can be learned during interacting with the environment, which fits well with optimization algorithms.f Therefore, this paper proposed a reinforcement learning-based ensemble particle swarm optimizer(RLEPSO), which combines reinforcement learning and EPSO and also has some improvements in other aspects.

The performance of proposed RLEPSO is tested on the CEC2013 test function set [10] with 28 functions. Moreover, compared with the other eight variants of particle swarm algorithms, the result shows that this algorithm has greatly surpassed existing PSO variant algorithms in the exploration and optimization capabilities.

## 2 PROPOSED RLEPSO

### 2.1 General idea

EPSO has achieved good results in optimization problems. However, as PSO variants become very complex, more and more running parameters need to be set, and the manual setting is troublesome and unable to get the best performance. Therefore, it is a better choice to obtain the optimal parameter set through learning.

What is more, because this parameter setting problem is essentially an optimization problem, and it is not derivable. There is no suitable loss function available, traditional deep learning algorithms cannot be applied to this optimization problem.

Therefore, this paper uses a reinforcement learning algorithm DDPG (deep deterministic policy gradient) to get a good action network to generate running parameters. This action network will guide the movement of particles in the optimization process according to their state. The complete RELPSO algorithm includes two parts: optimization rules and action network A. And to improve the algorithm's adaptability to different optimization goals, two algorithms, CLPSO and FDR-PSO, are implemented in RLEPSO.

### 2.2 Algorithm details

This paper will briefly introduce CLPSO and FDR-PSO, and will introduce the operation details of RLESO and the training of the action network in RLESO. This paper assumes that the readers have a basic understanding of PSO. In the following content, *pbest* is the particle's own best experience and *gbest* is the best experience in this swarm.

#### 2.2.1 *PSO variants employed in EPSO.*

*Comprehensive learning particle swarm optimizer (CLPSO).* In CLPSO, a particle learns from different particles' pbest for different dimensions. The velocity of $i_{th}$ particle is updated with the following equation:

$$V_i^d = wV_i^d + c * rand_i^d * (pbest_{fi(d)}^d - X_i^d) \tag{1}$$

In this equation, $f_i(d) = [f_i(1), f_i(2), ...., f_i(D)]$ defines which particle's pbest the $i_{th}$ particle should follow. CLPSO will set a $Pc$ value to determine which target one particle should follow. The target can be one particle's own pbest or other's pbest for each dimension $d$. Every particle have their own $Pc_i$. The $Pc_i$ value for each particle is generated by the following equation:

$$Pc_i = a + b * \frac{(exp(\frac{10(i-1)}{ps-1}) - 1)}{exp(10) - 1} \tag{2}$$

In this equation, ps is the population size, a = 0.05, b = 0.45. When a particle updates its velocity for one dimension, there will be a random value in [0,1] generated and compared with $Pc_i$. If the random value is larger than $Pc_i$, the particle of this dimension will follow its own pbest. Otherwise, it will follow another particle's pbest for that dimension. CLPSO will employ a tournament selection to choose a target particle. What's more, to avoid wasting function evaluations in the wrong direction, CLPSO defines a certain number of evaluations as refreshing gap $m$. During the period of a paticle following a target particle, the number of times the particle ceases improving is recorded as $flag_{clpso}$ If $flag_{clpso}$ is bigger than $m$,the particle will get his new target particle.

In order to combine CLPSO with our RLEPSO, we will define $V_{CLPSO}$ with following equation:

$$(V_{CLPSO})_i^d = rand_i^d * (pbest_{fi(d)}^d - X_i^d) \tag{3}$$

*Fitness-distance-ratio based particle swarm optimization (FDR-PSO).* Particles in FDR-PSO learn from their neighboring particle's experience (nbest) using a social learning component. Two criteria are proposed to choose a proper target particle: 1) the target particle must be near the particle being updated. 2) the target particle must be better than the particle being updated. To judge whether the target particles meet the requirements, the ratio of fitness distance to one-dimensional distance called Fitness-Distance-Ratio(FDR) is proposed. For $j_{th}$ particle, FDR is calculated using the following equation:

$$FDR = \frac{Fitness(X_i) - Fitness(X_j)}{|X_i - X_j|} \tag{4}$$

In this equation, $X_i$ denote the particle being updated, $X_j$ denote the target particle. In minimization problem, the particle which can minimize FDR is selected as the target particle. When the target particle is selected, the $i_{th}$ particle's velocity is updated using the following equation:

$$V_i^d = w * V_i^d + c1 * rand1_i^d * (pbest_i^d) + c2 * rand2_i^d * (gbest^d - X_i^d) + c3 * (nbest_j^d - X_i^d) \tag{5}$$

In this equation, $c1 = 1$, $c2 = 2$, $c3 = 2$. The nbest is the particle $X_j$ found by FDR. In order to combine FDR-PSO with our RLEPSO, we will define $V_{FDR}$ with following equation:

$$(V_{FDR})_i^d = rand_i^d * (nbest_j^d - X_i^d) \tag{6}$$

*2.2.2 RLEPSO Algorithm Optimization Process.* In EPSO, Self-adaptive selection strategy is proposed to select better PSO variant to solve the optimization problem. In order to make reinforcement learning available and to make the optimizing algorithm faster in RLEPSO, the self-adaptive selection strategy is removed and a combined velocity update equation is used. The equation is as follows:

$$\begin{aligned} V_{t+1} =& w * V_t + c1 * V_{CLPSO} + c2 * V_{FDR} + \\ & c3 * r1 * (gbest - X) + c4 * r2 * (pbest - X) \end{aligned} \tag{7}$$

In this equation, $V_{CLPSO}$ and $V_{FDR}$ are introduced in the previous section. pbest is the particle's own best experience, and gbest is the best experience in this swarm. According to the current running state, $w$, $c1$, $c2$, $c3$, and $c4$ are coefficients generated by the actor network. $r1$, $r2$ are all uniformly distributed random numbers between 0 and 1. To enhance the diversity of particles and increase the probability of finding the global optimal value, all the particles in RLEPSO will be divided into five swarms. Every swarm will have its own coefficients($w$, $c1$, and so on) but have the same $gbest$.

To prevent particles from being trapped in the local optimum, there is a mutation stage after the velocity updating. During this stage, first, a random number $r5$ between 0-1 will be generated, and then $r5$ will be compared with $C_{mutation} * 0.01 * flag_{clpso}$. If $r5$ is less than it, the mutation will be performed, and the particle position will be reinitialized in solution space.

At the end of one period, particles will move according to their velocity, and then particles' fitness and history best experience will be updated.

*2.2.3 actor network $\mu$.* In this paper, the action network $\mu$ is designed as a small network with 1-dimensional input and 35-dimensional output, with almost no additional computational cost. In each round of the particle swarm, the input

**Algorithm 1** RLEPSO algorithm.

```
Initialize the particle swarm and parameters
while fe ≤ femax do
    Caculate s_t {Eq.8}
    Caculate a_t with actor network μ(proposed in next subsubsection) and s_t
    for k = 1:n do
        Convert actions into operating parameters {Eq.9,10,11,12,13,14,15}
        Calculate the new speed {Eq.7}
        if randomvalue < C_mutation * 0.01 * flag_clpso then
            Reinitialize position
        else
            Update position x_{t+1} = x_t + v_t
        end if
        Calculate the evaluation value of all particles
        Update the parameters in particle operation
    end for
end while
```

content is recorded as the state vector $S_t$, and the output content is recorded as the action vector $A_t$. All the action value is between 0 and 1.

The state vector generation method is as follows:

$$S_t = fe_t / fe_{max} \tag{8}$$

$fe_t$ represents the number of function evaluations that have been executed in the t round, and $fe_{max}$ sets the number of function evaluations that need to be executed in this optimization process. The obtained action vector $A_t$ is 35-dimensional, divided into 5 groups, each of which is aimed at a sub-swarm. For a sub-swarm, the action vector is 7-dimensional a[0] to a[6]. The $w$, $c1$, $c2$, $c3$, $c4$, and $c_{mutation}$ required for each round of the optimization algorithm will be generated according to a[0] to a[6]. The generating formula is as follows:

$$c_{mutation} = a[0] * 0.01 * flag_{clpso} \tag{9}$$

$$w = a[1] * 0.8 + 0.1 \tag{10}$$

$$scale = 1/(a[3] + a[4] + a[5] + a[6] + 0.00001) * a[2] * 8 \tag{11}$$

$$c1 = scale * a[3] \tag{12}$$

$$c2 = scale * a[4] \tag{13}$$

$$c3 = scale * a[5] \tag{14}$$

$$c4 = scale * a[6] \tag{15}$$

*scale* is to control the range of speed change and prevent the particles from moving back and forth unstable.

*training algorithm :DDPG.* To train the actor network $\mu$, an reinforcement learning algorithm called deep deterministic policy gradient (DDPG)[14] is used. DDPG will be briefly introduced in the following content.

In a standard reinforcement learning environment, each agent interacts with the environment, and the ultimate goal is to maximize the benefits of the environment. This interactive process is described in a formatted manner as the Markov Decision Process (MDP), described by four-tuples (S, A, R, P). S is the state space, A is the action space,

$R : S \times A \rightarrow R$ is the reward function, and $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability. In this environment, an agent will learn a strategy $\pi : S \rightarrow$ A to maximize the environment's reward.

The action value function Q is generally used to represent the reward of performing a action in the s environment:

$$Q(s_t, a_t) = E[R_t | s = s_t, a = a_t] = E[\sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)] \tag{16}$$

In this equation, $r(s_i, a_i)$ represents the direct reward from current action. $Q(s_t, a_t)$ represents long-term reward from current action.

DDPG designs two deep neural networks, named the action value network $Q(s_t, a_t|\theta^Q)$ and the action network $\mu(s_t|\theta^\mu)$, where $\theta^Q$ and $\theta^\mu$ are the network parameters. The action network is a mapping corresponding to the state space and the action space, which can directly generate the required actions based on the state(It's actually strategy $\pi$). The action value network is used to approximate the action value function $Q$ and provide gradient for the action network's training.

The training of this action value network is to minimize the loss function:

$$L(\theta^Q) = (r(s_t, a_t) + \gamma Q'(s_{t+1}, a_{t+1}|\theta^Q) - Q(s_t, a_t|\theta^Q))^2 \tag{17}$$

$Q'$ is the target value network, and the weight is synchronized from network $Q$. The update of the action network parameters requires the use of the policy gradient algorithm, and the gradient update direction is as follows:

$$\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t, v)} = \nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t, v)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_t} \tag{18}$$

Through iteration, we can finally obtain a action network $\mu$.

*Training detail.* In the training process, each episode represents a complete optimization process of RPESO for the objective function. Each epoch represents a round of RPESO in the process of optimizing the objective function. Performing an action in the environment means inputting the generated action vector into RLESO, optimizing the target for one round, and obtaining the next round's reward and state.

The reward value of the algorithm is set as follows: When the best value of PSO in the current environment changes, the reward is 1. Otherwise, the reward is -1.

This setting is to improve the optimization speed of RLEPSO. The test function used to initialize the environment during each episode of training is randomly selected from CEC2013.

The pseudocode is proposed in Algorithm.2

## 3 EXPERIMENTS AND RESULTS

In this paper, the proposed RLEPSO algorithm's performance is evaluated using the shifted and rotated CEC2013 benchmark functions. The CEC2013 benchmark functions consist of 28 different types of unimodal, multimodal, expanded, and hybrid composition functions.

The algorithm for comparison includes a variety of variants of the PSO algorithm:

Inertia weight PSO (PSO), Comprehensive Learning PSO (CLPSO), Self-organizing hierarchical PSO with time varying acceleration coefficients (HPSO-TVAC), Fitness-Distance-Ratio based PSO (FDR-PSO), Distance-based locally informed PSO (LIPS), Orthogonal Learning PSO (OLPSO), Static Heterogeneous Swarm Optimization (sHPSO), Ensemble particle swarm optimizer(EPSO).

**Algorithm 2** train algorithm.

Randomly initialize $\theta^Q$ and $\theta^\mu$ in action network $\mu(s|\theta^\mu)$ and action value network $Q(s,a|\theta^Q)$.
Initialize the target network $Q'$ and $\mu'$, and its weight value is copied from $Q$ and $\mu$
Initialize the playback buffer R
**for** *episode* = 1 : *EpisodeMax* **do**
    Initialization environment (RLEPSO and evaluate function)
    **for** t=1, Tmax **do**t
        get observation $s_t$ from environment
        Choose actions based on $s_t$, network $\mu$ and explore noise
        Perform the action $a_t$ in the environment and observe the reward $r_t$ and the new state $s_{t+1}$
        Save ($s_t$, $a_t$, $r_t$, $s_{t+1}$) to the cache $R$
        Update Action Value Network by minimizing the loss function{Eq.17}
        Update the action network through the sampled action policy gradient:{Eq.18}
        Update the weights of the target network function

$$\theta^{Q'} = 0.9 * \theta^{Q'} + 0.1 * \theta^{Q}$$
$$\theta^{\mu'} = 0.9 * \theta^{\mu'} + 0.1 * \theta^{\mu}$$

    **end for**
**end for**

These algorithms have been introduced in the previous section. The first five algorithms are the popular ones in development, and the latter three algorithms are state-of-the-art algorithms in the field of particle swarm variants. The settings of these algorithms are the same in [16].

All experiments were carried out ten times, and the average value obtained was used as the final result. To test the optimization algorithm's adaptability in different situations, the test includes three different dimensions of 50, 30, and 10.

Table 1 is the result of the solution finally obtained by each function when the dimension is 50. In this table,The first place result is marked in bold, and the second place result is blue. RLEPSO ranks among the top four in all functions. In 11 functions RLEPSO have achieved first place results. In 10 functions RLEPSO have achieved second place results, and the worst ranking of RLEPSO is also located in fourth place. This shows that RLPSO has a solid ability to find optimization and ranks among the best in various evaluation functions. Moreover, it shows that RLESO is very stable, can adapt to various complex optimization goals, and can obtain a more stable solution.

Table 2 is the final average ranking of different algorithms in each dimension. It can be seen that whether it is dimension 50, 30, or 10, RLEPSO leads by absolute advantage. The average ranking at Dimension 50 surpasses the second one by 0.68. The average ranking at Dimension 30 surpasses the second one by 0.61. The average ranking at Dimension 10 surpasses the second one by 0.43. In the comprehensive average ranking of all dimensions, it is 0.6 ahead of the second one, which fully reflects the superiority of RLESO in terms of solution accuracy.

However, it can also be seen that since the training process is performed when the dimension is 50, there is a certain degree of lead reduction in the experiments of dimension 30 and dimension 10, which shows that this training method has a certain problem-solving relevance. In specific applications, one deployment of an algorithm is generally to solve a specific problem. Therefore, it is feasible to pre-train for a specific problem and then deploys it, and the effect is very significant.

In summary, RLEPSO has higher optimization accuracy than other similar algorithms and can effectively solve high-dimensional complex numerical optimization problems. The swarm optimization algorithm based on reinforcement

Table 1. Comparison of experimental results of PSO algorithms for 50 dimensional CEC2013 test functions. Functions.

| | CLPSO | EPSO | FDRPSO | HPSO-TVAC | LIPS | OLPSO | PSO | RLEPSO | SHPSO |
|---|---|---|---|---|---|---|---|---|---|
| F1 | 6.0E+04 | **-1.2E+03** | 6.0E+04 | 5.3E+04 | 7.9E+04 | 4.3E+04 | 2.8E+04 | -2.7E+02 | -9.3E+02 |
| F2 | 1.4E+09 | 8.9E+07 | 7.9E+08 | 6.1E+08 | 3.8E+09 | 2.2E+09 | 3.3E+08 | **7.5E+07** | 1.9E+08 |
| F3 | 5.5E+14 | **3.6E+10** | 3.7E+12 | 1.1E+12 | 7.1E+16 | 2.1E+14 | 2.1E+11 | 6.7E+10 | 1.3E+11 |
| F4 | 1.8E+05 | **7.9E+04** | 2.3E+05 | 8.3E+04 | 2.6E+05 | 3.9E+05 | 2.0E+05 | 1.1E+05 | 1.1E+05 |
| F5 | 2.5E+04 | **-8.4E+02** | 2.4E+04 | 6.3E+03 | 2.7E+04 | 2.1E+04 | 1.1E+04 | -6.9E+02 | -1.4E+02 |
| F6 | 5.6E+03 | **-6.3E+02** | 5.7E+03 | 3.8E+03 | 1.1E+04 | 4.1E+03 | 7.3E+02 | -6.1E+02 | -5.9E+02 |
| F7 | 5.1E+03 | **-6.5E+02** | -1.4E+02 | -1.5E+02 | 1.4E+05 | 7.4E+03 | -4.1E+02 | -6.4E+02 | -5.6E+02 |
| F8 | -6.8E+02 | -6.8E+02 | -6.8E+02 | -6.8E+02 | -6.8E+02 | -6.8E+02 | **-6.8E+02** | -6.8E+02 | -6.8E+02 |
| F9 | -5.2E+02 | -5.4E+02 | -5.2E+02 | -5.3E+02 | -5.2E+02 | -5.2E+02 | -5.4E+02 | **-5.5E+02** | -5.3E+02 |
| F10 | 9.6E+03 | **-1.1E+02** | 7.8E+03 | 6.0E+03 | 1.3E+04 | 9.9E+03 | 3.4E+03 | 2.2E+02 | 5.4E+02 |
| F11 | 7.5E+02 | 9.8E+01 | 7.8E+02 | 5.3E+02 | 8.4E+02 | 5.7E+02 | 2.1E+02 | -8.2E+01 | **-1.1E+02** |
| F12 | 9.9E+02 | 2.3E+02 | 9.2E+02 | 7.0E+02 | 1.0E+03 | 9.5E+02 | 5.0E+02 | **1.6E+02** | 3.6E+02 |
| F13 | 1.0E+03 | 3.6E+02 | 1.1E+03 | 7.8E+02 | 1.2E+03 | 1.1E+03 | 6.2E+02 | **3.4E+02** | 5.3E+02 |
| F14 | 1.1E+04 | 1.1E+04 | 1.4E+04 | 1.5E+04 | 1.6E+04 | 1.2E+04 | 9.3E+03 | **7.7E+03** | 8.0E+03 |
| F15 | 1.5E+04 | 1.5E+04 | 1.5E+04 | 1.6E+04 | 1.7E+04 | 1.6E+04 | 1.5E+04 | 1.4E+04 | **1.3E+04** |
| F16 | 2.0E+02 | 2.0E+02 | 2.0E+02 | 2.0E+02 | 2.1E+02 | 2.1E+02 | 2.0E+02 | 2.0E+02 | **2.0E+02** |
| F17 | 2.2E+03 | 8.6E+02 | 3.2E+03 | 1.5E+03 | 1.8E+03 | 2.8E+03 | 1.3E+03 | **7.2E+02** | 8.8E+02 |
| F18 | 2.4E+03 | 1.1E+03 | 3.3E+03 | 1.6E+03 | 1.9E+03 | 3.1E+03 | 1.5E+03 | **9.5E+02** | 1.2E+03 |
| F19 | 1.7E+06 | **6.0E+02** | 1.9E+06 | 2.1E+05 | 1.5E+06 | 4.4E+05 | 5.2E+05 | 7.2E+02 | 4.0E+04 |
| F20 | 6.2E+02 | 6.2E+02 | 6.2E+02 | 6.2E+02 | 6.2E+02 | 6.2E+02 | 6.2E+02 | **6.2E+02** | 6.2E+02 |
| F21 | 6.7E+03 | 1.9E+03 | 8.1E+03 | 4.9E+03 | 5.3E+03 | 1.2E+04 | 3.7E+03 | **1.7E+03** | 2.6E+03 |
| F22 | 1.4E+04 | 1.4E+04 | 1.7E+04 | 1.7E+04 | 1.8E+04 | 1.5E+04 | 1.1E+04 | **9.6E+03** | 1.0E+04 |
| F23 | 1.7E+04 | 1.7E+04 | 1.7E+04 | 1.7E+04 | 1.8E+04 | 1.7E+04 | 1.6E+04 | 1.6E+04 | **1.4E+04** |
| F24 | 1.5E+03 | 1.4E+03 | 1.4E+03 | 1.5E+03 | 2.1E+03 | 1.4E+03 | **1.4E+03** | 1.4E+03 | 1.4E+03 |
| F25 | 1.6E+03 | 1.5E+03 | 1.5E+03 | 1.6E+03 | 1.8E+03 | 1.5E+03 | 1.5E+03 | 1.5E+03 | **1.5E+03** |
| F26 | 1.7E+03 | **1.6E+03** | 1.7E+03 | 1.6E+03 | 1.8E+03 | 1.7E+03 | 1.6E+03 | 1.6E+03 | 1.7E+03 |
| F27 | 3.9E+03 | 3.4E+03 | 3.5E+03 | 3.9E+03 | 4.9E+03 | 3.7E+03 | **3.2E+03** | 3.2E+03 | 3.3E+03 |
| F28 | 1.0E+04 | 3.5E+03 | 8.7E+03 | 9.1E+03 | 1.3E+04 | 1.2E+04 | 6.3E+03 | **2.7E+03** | 4.1E+03 |

Table 2. Average rank of RLEPSO on other dimentions.

| | RLEPSO | CLPOS | EPSO | FDRPSO | HPSOTVAC | LIPS | OLPSO | PSO | SHPSO |
|---|---|---|---|---|---|---|---|---|---|
| 50D | 1.93 | 6.50 | 2.61 | 6.64 | 5.43 | 8.54 | 7.21 | 3.43 | 2.71 |
| 30D | 1.93 | 6.29 | 2.89 | 6.18 | 6.25 | 8.61 | 6.32 | 4.00 | 2.54 |
| 10D | 2.07 | 5.21 | 3.14 | 6.93 | 5.71 | 8.89 | 6.57 | 3.96 | 2.50 |
| ALL | 1.98 | 6.00 | 2.88 | 6.58 | 5.80 | 8.68 | 6.70 | 3.80 | 2.58 |

learning has better optimization and exploration capabilities than the swarm optimization algorithm with manually set parameters.

## 4 CONCLUSION

This paper proposes an ensemble particle swarm optimization algorithm based on reinforcement learning, integrating two particle swarm algorithm variants, CLPSO and FDR-PSO. Besides, the algorithm uses the reinforcement learning algorithm DDPG to train an action network A, which efficiently provides the algorithm's running parameters according

to the current stage. Also, to enhance the algorithm's global search capabilities, all particles will be divided into 5 sub-swarms. Each sub-swarm will independently have the running parameters from actor network A. Moreover, the algorithm adds a new mutation step in the movement process to prevent the particles from being trapped into the local optimal value prematurely. During this period, particles will reinitialize randomly according to the algorithm's current configuration and the number of times the particles stop growing. In this way, the population diversity and global search capabilities of RLPSO are excellent.

The RLEPSO algorithm's performance is tested through the CEC2013 standard test set and compared with other 8 particle swarm variants. The results showed that RLPSO performed well on all test functions and achieved the top two results on 21 functions out of 28 functions. RLEPSO outperforms its individual PSO variants as well as recent state-of-the-art PSO algorithms. The future research directions for RLEPSO include adding other optimizing algorithms into RLEPSO and using DDPG on other optimization algorithms.

## REFERENCES


[1] Swagatam Das and Ponnuthurai N Suganthan. 2010. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. *Jadavpur University, Nanyang Technological University, Kolkata* (2010), 341–359.

[2] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. *IEEE computational intelligence magazine* 1, 4 (2006), 28–39.

[3] Andries P. Engelbrecht. 2010. Heterogeneous particle swarm optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6234 LNCS (2010), 191–202. https://doi.org/10.1007/978-3-642-15461-4_17

[4] Vitaliy Feoktistov. 2006. *Differential evolution.* Springer.

[5] Bo He, Lulu Ying, Shujing Zhang, Xiao Feng, Tianhong Yan, Rui Nian, and Yue Shen. 2015. Autonomous navigation based on unscented-FastSLAM using particle swarm optimization for autonomous underwater vehicles. *Measurement* 71 (2015), 89–101.

[6] Wenbin Hu, Huan Wang, Zhenyu Qiu, Cong Nie, and Liping Yan. 2018. A quantum particle swarm optimization driven urban traffic light scheduling model. *Neural Computing and Applications* 29, 3 (2018), 901–911.

[7] Dervis Karaboga. 2010. Artificial bee colony algorithm. *scholarpedia* 5, 3 (2010), 6915.

[8] TH Kim, I Maruta, and T Sugie. 2010. A simple and efficient constrained particle swarm optimization and its application to engineering design problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 224, 2 (2010), 389–400.

[9] JJ Liang, BY Qu, PN Suganthan, and Q Chen. 2014. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* 29 (2014), 625–640.

[10] JJ Liang, BY Qu, PN Suganthan, and Alfredo G Hernández-Díaz. 2013. 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 (2013), 281–295.

[11] JJ Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. 2006. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics* 41, 8 (2006), 8–31.

[12] Jing J. Liang, A. K. Qin, Ponnuthurai Nagaratnam Suganthan, and S. Baskar. 2006. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation* 10, 3 (2006), 281–295. https://doi.org/10.1109/TEVC.2005.857610

[13] Jing J Liang, Bo Y Qu, and Ponnuthurai N Suganthan. 2013. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* 635 (2013), 490.

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* (2016). arXiv:1509.02971

[15] Jianshan Lu. 2017. System identification of force transducers for dynamic measurements using particle swarm optimization. *Journal of Vibroengineering* 19, 2 (2017), 864–877.

[16] Nandar Lynn and Ponnuthurai Nagaratnam Suganthan. 2017. Ensemble particle swarm optimizer. *Applied Soft Computing Journal* 55 (2017), 533–548. https://doi.org/10.1016/j.asoc.2017.02.007

[17] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. 2010. Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore* 24 (2010).

[18] Seyedali Mirjalili. 2019. Genetic algorithm. In *Evolutionary algorithms and neural networks.* Springer, 43–55.

[19] Maroua Nouiri, Abdelghani Bekrar, Abderezak Jemai, Smail Niar, and Ahmed Chiheb Ammari. 2018. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing* 29, 3 (2018), 603–615.

[20] T. Peram, K. Veeramachaneni, and C. K. Mohan. 2003. Fitness-distance-ratio based particle swarm optimization. *2003 IEEE Swarm Intelligence Symposium, SIS 2003 - Proceedings* 2 (2003), 174–181. https://doi.org/10.1109/SIS.2003.1202264

[21] Joram Piatigorsky and Graeme J Wistow. 1989. Enzyme/crystallins: gene sharing as an evolutionary strategy. *Cell* 57, 2 (1989), 197–199.

[22] B. Y. Qu, Ponnuthurai Nagaratnam Suganthan, and Swagatam Das. 2013. A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 387–402. https://doi.org/10.1109/TEVC.2012.2203138

[23] Asanga Ratnaweera, Saman K. Halgamuge, and Harry C. Watson. 2004. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation* 8, 3 (2004), 240–255. https://doi.org/10.1109/TEVC.2004.826071

[24] KV Shihabudheen, M Mahesh, and GN Pillai. 2018. Particle swarm optimization based extreme learning neuro-fuzzy system for regression and classification. *Expert Systems with Applications* 92 (2018), 474–484.

[25] Ponnuthurai N Suganthan, Nikolaus Hansen, Jing J Liang, Kalyanmoy Deb, Ying-Ping Chen, Anne Auger, and Santosh Tiwari. 2005. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL report* 2005005, 2005 (2005), 2005.

[26] Xiaoping Yang, Xueying Chen, Riting Xia, and Zhihong Qian. 2018. Wireless sensor network congestion control based on standard particle swarm optimization and single neuron PID. *Sensors* 18, 4 (2018), 1265.

[27] Xin Yao, Yong Liu, and Guangming Lin. 1999. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation* 3, 2 (1999), 82–102.

[28] Zhi Hui Zhan, Jun Zhang, Yun Li, and Yu Hui Shi. 2011. Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 15, 6 (2011), 832–847. https://doi.org/10.1109/TEVC.2010.2052054