An Adaptive Online Parameter Control Algorithm for Particle Swarm Optimization Based on Reinforcement Learning

1st Yaxian Liu
School of Electronic and Information Engineering
Beihang University
Beijing, China
rexliuyx@buaa.edu.cn

2nd Hui Lu
School of Electronic and Information Engineering
Beihang University
Beijing, China
mluhui@buaa.edu.cn

3rd Shi Cheng
School of Computer Science
Shaanxi Normal University
Xi'an, China
cheng@snnu.edu.cn

4th Yuhui Shi
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
shiyh@sustc.edu.cn

***Abstract*—Parameter control is critical to the performance of any evolutionary algorithm (EA). In this paper, we propose a Q-Learning-based Particle Swarm Optimization (QLPSO) algorithm, which uses the Reinforcement Learning (RL) to train the parameters in Particle Swarm Optimization (PSO) algorithm. The core of the QLPSO algorithm is a three-dimensional Q table which consists of a state plane and an action axis. The state plane includes the state of the particles in both of the decision space and the objective space. The action axis controls the exploration and exploitation of particles by setting different parameters. The Q table can help particles to select actions according to their states. Besides, the Q table should be updated by reward function which is designed according to the performance change of particles and the number of iterations. The main difference between the QLPSO algorithms for single-objective and multi-objective optimization lies in the evaluation of the solution performance. In single-objective optimization, we only compare the fitness values of solutions, while in multi-objective optimization, we need to discuss the dominant relationship between solutions with the help of Pareto front. The performance of QLPSO is tested based on 6 single-objective and 5 multi-objective benchmark functions. The experiment results reveal the competitive performance of QLPSO compared with other algorithms.**

***Index Terms*—optimization problem, parameter control, particle swarm optimization, reinforcement learning**

I. Introduction

Evolutionary algorithms (EAs) play an imperative role in the field of optimization. When solving different optimization problems, appropriate parameters need to be configured for EAs. However, the performance of EAs greatly depends on the parameter values and they should even change along with the run of EAs, which shows the importance of the online parameter control problem. Besides, online parameter control is also very challenging because it is hard to determine the strategy to monitor the running state of EAs and then adjust the parameters. Here, we will focus on online parameter control and design an adaptive parameter control algorithm for the widely used Particle Swarm Optimization (PSO) algorithm.

In this paper, we propose a Q-Learning-based PSO (QLPSO) algorithm by embedding the Reinforcement Learning (RL) into the process of the adaptive online parameter control of PSO. In the proposed algorithm, we train the Q table to help particles select their optimal actions (parameters) according to their states. The Q table has three dimensions: the decision space state, the objective space state and the action. The decision space state reflects the distance of positions between a particle and the optimal solution. The objective space state concerns the difference of values between a particle and the optimal solution. The action controls the exploration and the exploitation of particles by setting different parameters. All particles share one Q table and each particle can contribute to the update of the Q table. Reward function is needed when updating the Q table. The design of reward function considers both the evaluation of particles performance and the influence of generation. The process of utilizing and updating Q table for each particle is explained as follows. First, the states of the particle in both decision space and objective space are determined. Second, each particle will select the action with the maximal Q value under this state. Then, the performance of the particle before and after the action will be compared and evaluated. Finally, the Q table will be updated according to the reward function.

There are two versions of the QLPSO algorithm named as QLSOPSO and QLMOPSO, which are used to solve single-objective and multi-objective optimization problems, respectively. The main difference between QLSOPSO and QLMOP-

This research is supported by the National Natural Science Foundation of China under Grant No. 61671041, No.61101153 and No. 61761136008.

978-1-7281-2153-6/19/$31.00 ©2019 IEEE

SO lies in the method to evaluate a particles performance. In the single-objective optimization, we only need to compare the fitness values of particles, while in the multi-objective optimization, the dominant relationship between solutions need to be discussed with the help of Pareto front. The difference is embodied in the determination of objective space state and the design of reward function, which will be elaborated in Section 3. In addition, a set of experiments are conducted to test the parameter control effect of the QLSOPSO and QLMOPSO based on 6 single-objective and 5 multi-objective benchmark functions. Experiment results show that the QLPSO algorithms have a better performance compared with other methods.

The rest of the paper is organized as follows. Section 2 gives a summary of recent researches for the parameter control problem. Section 3 briefly introduces the principle of single-objective PSO, multi-objective PSO and Q-Learning. The proposed QLPSO algorithm is explained in Section 4. The experiments and statistic results are presented and discussed in Section 5. Section 6 concludes the paper and points out further research suggestions.

II. Related Work

There are a lot of works in the literature on online parameter control. The mechanisms of them can be clearly classified into three categories: deterministic, self-adaptive and adaptive.

Deterministic methods follow a preset rule for assigning new parameter values, which is often associated with the running time without any feedback from the search. Two kinds of PSO algorithm, respectively equipped with the linear and nonlinear decreasing inertia weight mechanisms, were proposed successively [1,2]. Ratnaweera et al. further proposed a PSO variant with linear time-varying cognition acceleration coefficients [3]. This kind of method is easy to implement, but the fixed parameter adjustment strategy is not bound to be suitable for all problems.

Self-adaptive parameter control combines the search of optimal parameters with the search of optimal solutions. For example, in the algorithm proposed by Li et al., the inertial weight and two cognition acceleration coefficients are served as the last three components of the position vector of the particles, so that the parameters can co-evolve with the particles [4]. Eiben et al. self-adapted the selective pressure in Genetic Algorithm (GA) by encoding the tournament size in the genome [5]. However, the implicit parameter adjustment incorporated in the algorithm is not conducive to understanding and analysis.

Adaptive parameter control will monitor the performance when the EA is running. The change of the performance is used as a feedback to guide the parameter control. The model for adaptive parameter control is proposed in [6]. Iwasaki et al. proposed a linear decreasing ideal velocity curve to guide the parameter adjustment of PSO [7]. Based on the above-mentioned algorithm, Xu adopted the non-linear decreasing ideal velocity curve, which further improved the performance of PSO to some degree [8]. Shing et al. proposed an adaptive parameter control algorithm for PSO based on historical data. If the slope of average fitness began to decrease, the parameters of PSO would be adjusted based on the binary space partitioning tree [9]. Adaptive method adjusts the parameter in an explicit way. Nevertheless, it is hard to confirm the direction and degree of the parameter adjustment.

Apart from that, the combination of Reinforcement Learning (RL) and optimization algorithms is not new in this field. Grigoris et al. proposed a hybrid approach that integrates the PSO algorithm with the RL approach to efficiently tackle noisy problems[10]. In the Q-learning-based swarm optimization algorithm proposed in [11], the best individual is chosen based on its accumulated performance instead of its momentary performance at each evaluation. Jamal et al. used RL to determine value of effective parameters of the variable neighborhood search (VNS) according to dynamic job shop scheduling (DJSS) problem situation dynamically [12]. However, few attempts have been made to adaptively adjust the parameters of PSO algorithm by RL.

Granted that there are a lot of promising methods, parameter control for EAs still remains challenging and unsolved considering the following problems and obstacles. First, the parameters in EAs not only should be varied according to different optimization problems but also should change with the run of algorithms. Second, strictly speaking, parameter adaptive control has not been achieved actually, because the mechanism to determine the magnitude of adjustment is still deterministic. Third, how to avoid introducing new parameters and reduce the time and space complexity is also a problem worth thinking about. As a result, the existing parameter control methods are still not satisfactory so far.

III. Background Information

A. Particle Swarm Optimization

PSO was introduced by Kennedy and Eberhart in 1995 [13]. The motivation of PSO is to mimic the foraging activity of the birds. It was originally devised to solve the single-objective optimization problem, while in 2002 Coello and Lechuga modified the original PSO algorithm and proposed multi-objective PSO (MOPSO) algorithm to deal with multi-objective optimization problems [14].

1) Single-Objective PSO: In general, PSO is represented by a swarm of N particles. All particles in the swarm are associated with two vectors to describe the state of themselves, namely velocity V and position X, which are defined in (1) and (2) below, where D represents the dimension of the search space.

$$V_i = (V_i^1, V_i^2, ..., V_i^D), i = 1, 2, ..., N \tag{1}$$

$$X_i = (X_i^1, X_i^2, ..., X_i^D), i = 1, 2, ..., N \tag{2}$$

All particles are equipped with the memory ability, which enables them to remember the local best positions Pb achieved by particles. Also, all the Pb of different particles will be shared among the whole swarm, and the optimal one is regarded as the global best position Gb achieved by the swarm.

Then all particles in the swarm will update their velocities and positions according to Pb and Gb using (3) and (4) below [13].

$$V_{id}^{t+1} = \omega \times V_{id}^{t} + c_1 \times rand(0,1) \times (Pb_{id}^{t} - X_{id}^{t}) + c_2 \times rand(0,1) \times (Gb_{d}^{t} - X_{id}^{t}) \quad (3)$$

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t} \quad (4)$$

Here, ω is the inertia weight, c_1 is the cognitive acceleration coefficient, c_2 is the social acceleration coefficient, rand(0,1) is a uniformly distributed random number within [0,1], and V_{id}^{t} denotes the d^{th} dimensional velocity of the i^{th} particle in the t^{th} generation.

2) Multi-Objective PSO: In the multi-objective optimization, it is more difficult to evaluate particles, since the optimal value of each objective can not be satisfied at the same time. Therefore, the selection of the local best position Pb and the global best position Gb will be determined according to the Pareto front and the crowding distance [15]. Specifically, when there is a dominant relationship between two solutions, which one is better is decided according to Pareto front. On the other hand, if the two solutions are on the same Pareto front, the optimal solution is the one with smaller crowding distance. Moreover, in MOPSO, the non-dominated solutions obtained during the searching process will be saved in an archive. The strategy used to update the velocity and position of the particles is the same as (3) and (4). The comparison between single-objective PSO and multi-objective PSO is shown in Fig. 1.

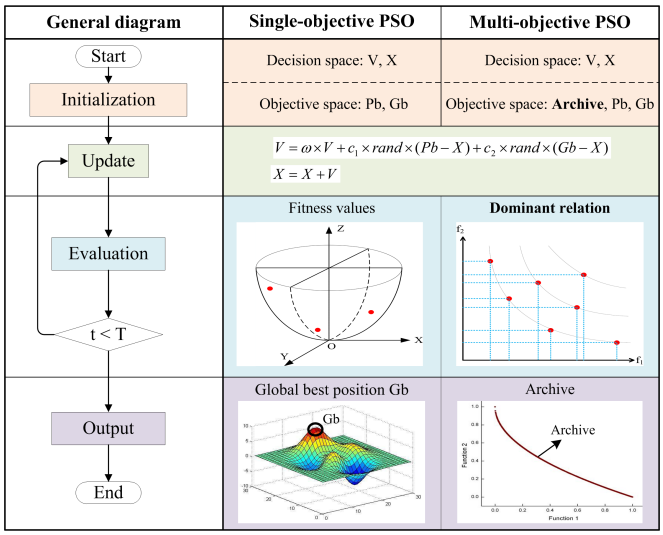


Fig. 1. The comparison between single-objective and multi-objective PSO.

B. Q-Learning

Reinforcement learning (RL) is a kind of machine learning concerned with how agents ought to take actions in an environment so as to maximize the accumulative reward. There are three key elements in RL: state, action and reward [16]. Q-learning is a kind of RL proposed by Watkins in 1989 [17]. Q-learning consists of a Q table which can guide the agent to choose the optimal action with the maximal Q value under a certain state. Also, the Q table needs to be updated in the process of training. The update strategy is shown in (5).

$$Q(s_{t+1}, a_{t+1}) = (1-\alpha)Q(s_t, a_t) + \alpha[R(s_t, a_t) + \gamma \max_{a} Q(s_{t+1}, a)] \quad (5)$$

Here, α is the learning rate, γ is the discount factor, $R(s_t, a_t)$is the immediate reward acquired from executing action a_t under the state s_t, and $Q(s_t, a_t)$ is the accumulative reward that the agent has gained at time t. The model of Q-leaning is shown in Fig. 2.

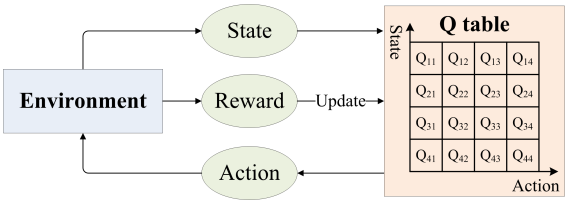


Fig. 2. The model of Q-leaning.

IV. QLPSO Algorithm

QLPSO algorithm can achieve the online adaptive parameter control in PSO based on the Q-learning. The four important elements of the QLPSO algorithm are state, action, Q table and reward, which are shown in Fig. 3. Their relationship is that the Q table, which can be updated by the reward function, helps a particle to determine the action according to its state. In this section, we will introduce the QLPSO algorithm in terms of the selection of state and action, the design of Q table and the determination of reward function.

A. State and Action

To begin with, we need to decide the state of the particle in the decision space and the objective space. In the decision space, we define 4 kinds of states: nearest, nearer, farther, farthest, which respectively represent the relative distance between the particle and the global best position Gb compared with the size of search space. In the objective space, we also define 4 kinds of states: smallest, smaller, larger, largest, which respectively represent the relative performance of the particle compared with the global best fitness and the global worst fitness. However, since the methods for evaluating the performance of a particle are different in single-objective PSO and multi-objective PSO, we need to discuss them separately in the objective space. In terms of the single-objective PSO, we just need to consider and compare the fitness value between two solutions. As for the multi-objective PSO, we should firstly sort all of the particles based on Pareto front and crowding distance, and then distribute the state for the particles according to the order of the whole population. The

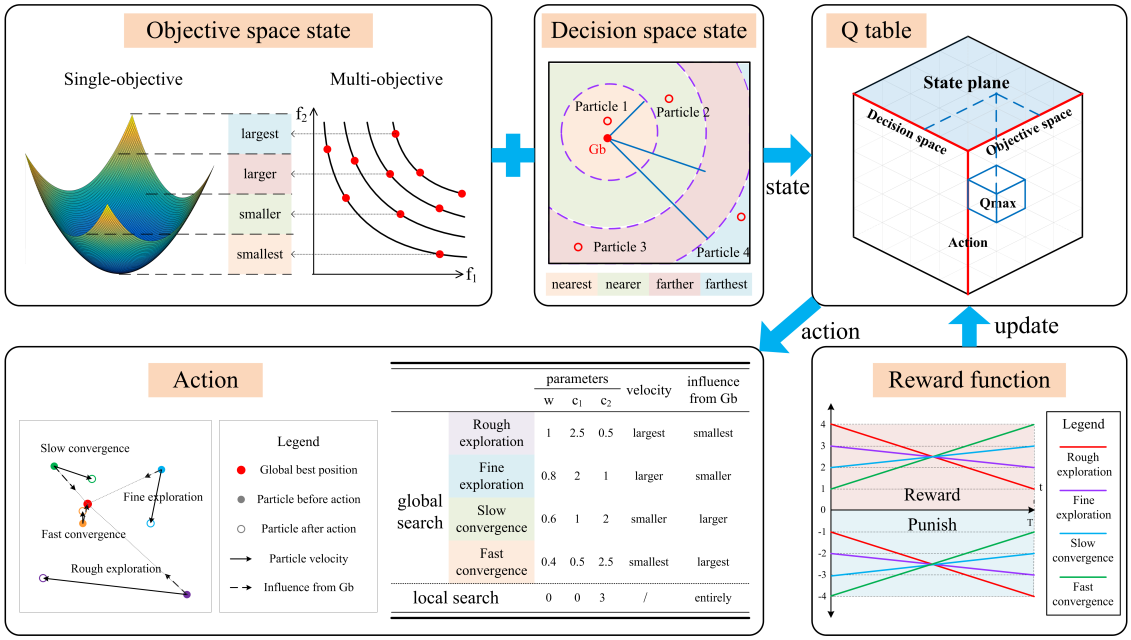


Fig. 3. The important elements in the QLPSO algorithm.

TABLE I
THE DEFINITION OF DECISION SPACE STATE AND OBJECTIVE SPACE STATE.

Relative distance (d)	Decision space state	Single-objective problem Relative fitness (f)	Multi-objective problem Ordinal number (i)	Objective Space State
$0 \leq d \leq 0.25\Delta R$	Nearest	$0 \leq f \leq 0.25\Delta F$	$0 \leq i \leq 0.25N$	Smallest
$0.25\Delta R \leq d \leq 0.5\Delta R$	Nearer	$0.25\Delta F \leq f \leq 0.5\Delta F$	$0.25N \leq i \leq 0.5N$	Smaller
$0.5\Delta R \leq d \leq 0.75\Delta R$	Farther	$0.5\Delta F \leq f \leq 0.75\Delta F$	$0.5N \leq i \leq 0.75N$	Larger
$d \geq 0.75\Delta R$	Farthest	$f \geq 0.75\Delta F$	$i \geq 0.75N$	Largest

definitions of decision space state and objective space state are respectively shown in Table 1.

Here, d is the Euclidean distance between a certain particle and the global best position Gb. ΔR is the range of decision space, which refers to the difference between the upper bound and the lower bound of the decision space. f is the relative fitness, which refers to the difference of fitness between a certain particle and the global best position Gb. ΔF is the scope of objective space, which represents the difference between the global best fitness and the global worst fitness. i is the ordinal number of a certain particle in the sort determined by the Pareto Front and the crowding distance. N represents the size of the population.

Apart from that, we classify the actions of particles into two categories: {global search, local search}. In the first 90% of the generation, the population performs global search to avoid the particles falling into the local optimum. While in the last 10% of the generation, the population performs local search to find a more accurate optimal solution. A lot of experiments show that the time allocation of global search and local search has desirable robustness. As long as the time of local search accounts for 5% to 20% of the total number of iterations, the QLPSO algorithm will have a satisfactory performance in finding the optimal solution. Global search contains four classes: {rough exploration, fine exploration, slow convergence, fast convergence}. Each action corresponds to a set of parameters.

B. Q table

Now, we need to design the Q table which is used to decide the action of each particle according to its state. Since the decision space and the objective space are both indispensable in the optimization problem, the Q table in the QLPSO algorithm is different from the two-dimensional one which is widely used in the traditional Q-learning. In the QLPSO algorithm, we design a three-dimensional Q table in order to take both the decision space and the objective space into consideration. The process of choosing the action of a particle based on the state is explained as follows. First, the state of a particle in the decision space and the objective space enables us to determine one certain location in the state plane, for example, {farther, smaller}. Second, the Q column which is associated with the location in the state plane will be extracted.

Third, we choose the action which has the maximal Q value in the Q column for the particle.

C. Reward Function

Two problems need to be considered in the design of reward function: 1) how to determine whether a particle should be rewarded or punished? 2) how to determine the degree of reward and punishment?

In terms of the first problem, the main idea is explained as follows. After executing a certain action, if the performance of the particle is better than before, the corresponding Q value should increase. On the contrary, if the performance of the particle becomes worse after the action, the corresponding Q value should decrease. Since the performance of each particle is involved, we need to discuss single-objective PSO and multi-objective PSO separately. In the single-objective PSO, if the fitness of a particle is superior to before, we should reward the action. Conversely, if the fitness becomes worse or remain unchanged, the action should be punished. In the multi-objective PSO, if the particle after action dominates the particle before action, we should reward the action. However, if the particle after action is dominated by the particle before action or there is no dominant relation between them, the action should be punished.

As for the degree of reward and punishment, the influence of running phase of algorithm should be taken into account. At the beginning of the algorithm, the exploration should be encouraged so as to avoid falling into the local optimum. On the contrary, at the end of the algorithm, the convergence should be promoted in order to find out a more accurate optimal solution. Therefore, we design the reward function which is shown in Fig. 3.

D. Summary

The flowchart of QLPSO algorithm is shown in the Fig. 4. First, initialize the population and Q table. Second, decide the state of each particle in objective space and decision space according to its position. Third, determine the action (parameter) of the particle using the Q table. Fourth, update the particle based on the parameters determined in the previous step. Next, update the Q table based on the reward function. Then, repeat these steps for all particles in each generation until the number of iteration reaches 90% of the maximum. Finally, stop global search and start local search until the end of the algorithm.

The pseudo code of the QLPSO algorithm is given below.

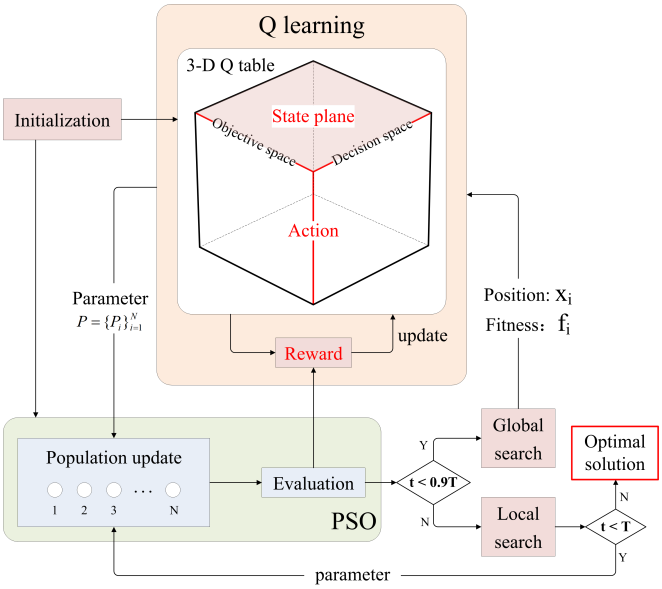


Fig. 4. The flowchart of QLPSO algorithm.

Algorithm 1 QLPSO

```
Initialize population and Q table
while t <= T
    if t <= 0.9T                    //global search
        while n <= N
            decide the state in two spaces
            determine the action using Q table
            update v and x
            evaluate the performance
            update Q table using reward function
            n++
        end while
    else                            //local search
        while n <= N
            update v and x
        end while
    end if
    t++
end while
```

V. Experiments

In order to verify the performance of the QLPSO algorithm, we design experiments from two aspects: single-objective PSO and multi-objective PSO. In each part, we compare the performance of QLPSO with the linear decreasing inertia weight PSO (LPSO) and the standard PSO (SPSO). The uppercase characters S, L, and QL represent the SPSO, LPSO and QLPSO, respectively. The parameter settings for all experiments are shown in Table 2. In LPSO algorithm, the range from 0.9 to 0.4 is a good area to choose inertia weight for PSO to get better performance, which is concluded form a large number of experiments in [18]. Therefore, we use the median value of this range to initialize the SPSO algorithm to ensure the fairness of comparative experiments.

A. Single-Objective PSO

We selected 6 benchmark functions (maximum problem) which are widely used in single-objective optimization to test the performance of the QLSOPSO algorithm. All of them have one global maximum and several local maxima. The details of these benchmark functions are shown in Fig. 5.

Here, we compare the performance of QLSOPSO with LSOPSO and SSOPSO. For each problem, all of the three

TABLE II
THE PARAMETER SETTINGS FOR ALL EXPERIMENTS.

Parameter setting	single-objective PSO			multi-objective PSO		
	S	L	QL	S	L	QL
Inertia weight	0.65	[0.9,0.4]	/	0.65	[0.9,0.4]	/
Experiment times	500			100		
Population size	50			20		
iteration times	100			40		
Dimension	2			2		
Function number	6			5		

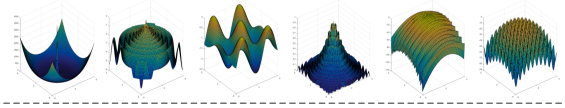
Name	Needle	Labyrinth	Griewank	Dropwave	Levy	Rastrigin
Image						
Range	[-5,5]	[-5,5]	[-10,10]	[-5,5]	[-5,5]	[-5,5]
Optimum	3600	0	0	1	0	0

Fig. 5. The single-objective benchmark functions.

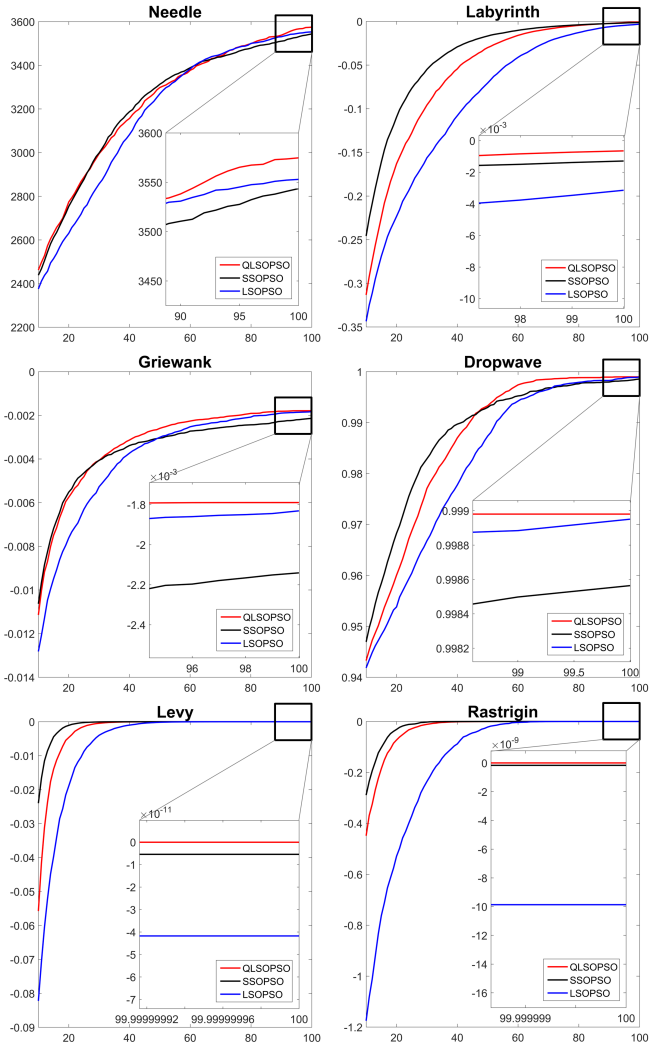


Fig. 6. The statistical results for solving single-objective optimization problems.

algorithms run for 500 times, and the statistical results are shown in Fig. 6. The horizontal axis represents the iteration times, and the vertical axis refers to the statistical average of fitness in a certain generation.

At the beginning of the algorithm, the increasing rate of fitness in QLSOPSO is a little slower than SSOPSO while faster than LSOPSO, because QLSOPSO focuses more on the diversity of particles in order to avoid falling into the local optimal solution. At the end of the algorithm, the optimal solutions obtained by QLSOPSO in 6 benchmark functions are all superior to other two methods. It indicates that the performance of the QLSOPSO is better than SSOPSO and LSOPSO when solving the single-objective optimization problem.

B. Multi-Objective PSO

We selected 5 benchmark functions (minimum problem) from ZDT function set which are widely used in multi-objective optimization to test the performance of the QLMOPSO algorithm. The details of these benchmark functions are shown in Table 3.

Here, we compare the performance of QLMOPSO with LMOPSO and SMOPSO. The distribution of the population in the last generation and the archive are shown in Fig. 7 and Fig. 8, respectively. The black lines represent the ideal Pareto fronts.

From Fig. 7, we can draw a conclusion that almost all the particles of the last generation in QLMOPSO are distributed on the Pareto front. However, the convergence of SMOPSO is comparatively poorer, and the particles in several functions are even trapped into local optimum. Although LMOPSO is better than SMOPSO to some degree because there are a little bit more particles located on the ideal Pareto front, the results are still not satisfactory.

From Fig. 8, there are still some particles which are not on the ideal Pareto front in the archive of SMOPSO and LMOPSO, especially in ZDT6. Besides, since there are some gaps in the front of SMOPSO and LMOPSO, the diversity of them is not as good as QLMOPSO, either. The problems of the fronts in SMOPSO and LSOPSO have been marked with green boxes in Fig. 8.

In order to evaluate and compare the performance of algorithms quantitatively and convincingly, several metrics are adopted in this paper, i.e., γ, GD, Δ, S and HV. Among them, γ and GD evaluate the convergence of the Pareto optimal set, Δ and S reflect the diversity of the Pareto optimal set, and HV is a comprehensive index considering both of these two

TABLE III
THE MULTI-OBJECTIVE BENCHMARK FUNCTIONS.

Name	Range	Objective function
ZDT1	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$ $g(x) = 1 + 9x_2$
ZDT2	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - (\frac{x_1}{g(x)})^2]$ $g(x) = 1 + 9x_2$
ZDT3	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)} sin(10\pi x_1)]$ $g(x) = 1 + 9x_2$
ZDT4	$x_1 \in [0, 1]$ $x_2 \in [-5, 5]$	$f_1(x) = x_1$ $f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$ $g(x) = 11 + x_2^2 - 10cos(4\pi x_2)$
ZDT6	$x_1 \in [0, 1]$ $x_2 \in [-5, 5]$	$f_1(x) = 1 - exp(-4x_1)(sin(6\pi x_1))^6$ $f_2(x) = g(x)[1 - (\frac{f_1(x)}{g(x)})^2]$ $g(x) = 1 + 9x_2^{0.25}$

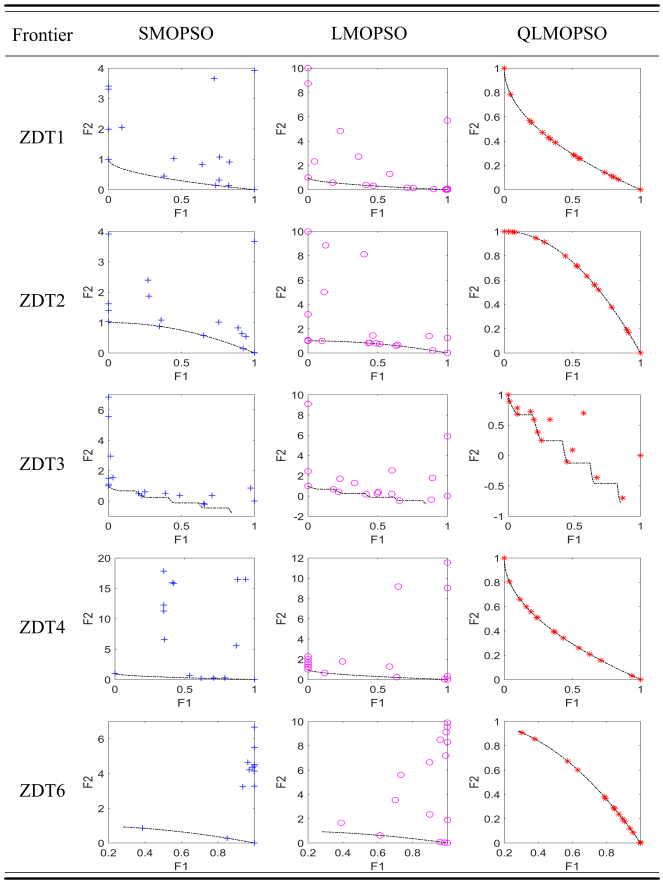


Fig. 7. The distribution of the population in the last generation.

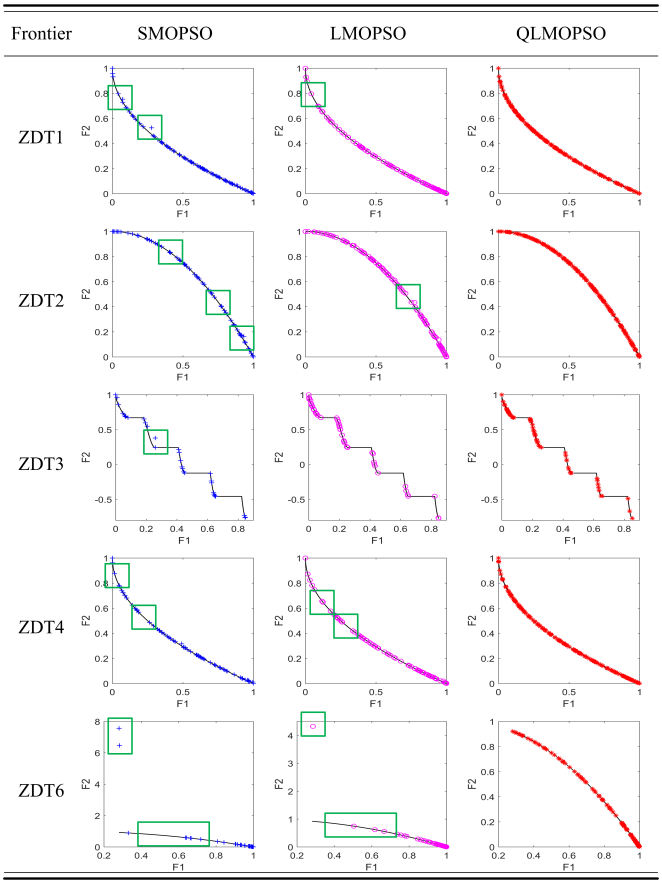


Fig. 8. The distribution of the archive.

aspects. The larger the HV, the better the performance of the algorithm. On the contrary, for the other four metrics, the smaller the better. The details about the precise definition of these performance metrics are explained in [19]. The mean values of performance metrics of different algorithms when dealing with various problems are shown in Table 4. In all of the cases, the best performances are denoted in bold.

The boxplots for the above results are shown in Fig. 9.

As for the convergence metric γ, the results show that SMOPSO is slightly better than other two methods which are equipped with different parameter control strategies. It is inevitable because LMOPSO and QLMOPSO are both aimed at increasing the diversity of particles at the beginning of the algorithm, which will affect the convergence to some degree. However, in terms of the metrics GD, Δ, S and HV, QLMOPSO always performs better than SMOPSO and LMOPSO. Therefore, we can safely draw a conclusion that QLMOPSO is superior to SMOPSO and LMOPSO when solving the multi-objective optimization problem.

VI. CONCLUSION

In this paper, a Q-Learning-based Particle Swarm Optimization (QLPSO) algorithm has been proposed for achieving a better performance of PSO by controlling the parameters.

TABLE IV
THE MEAN VALUES OF PERFORMANCE METRICS.

Mean		Convergence		Diversity		HV
		γ	GD	Δ	S	
ZDT1	S	**0.00022**	7.34E-5	1.53609	0.00993	0.86038
	L	0.00027	6.83E-5	1.46616	0.00867	0.86324
	QL	0.00025	**3.28E-5**	**1.15135**	**0.00562**	**0.87078**
ZDT2	S	**0.00020**	6.01E-5	1.54184	0.00913	0.53097
	L	0.00028	6.59E-5	1.44329	0.00777	0.53371
	QL	0.00023	**2.76E-5**	**1.14339**	**0.00539**	**0.53757**
ZDT3	S	**0.00033**	0.00012	1.72999	0.02407	1.00077
	L	0.00045	0.00015	1.69456	0.02360	0.99401
	QL	0.00043	**0.00011**	**1.40794**	**0.01952**	**1.01463**
ZDT4	S	**0.00020**	5.34E-5	1.52631	0.00981	0.86064
	L	0.00023	5.01E-5	1.42990	0.00853	0.86449
	QL	0.00026	**3.45E-5**	**1.14846**	**0.00562**	**0.87084**
ZDT6	S	0.01991	0.01474	1.54805	0.15134	0.40043
	L	0.02321	0.01772	1.50991	0.19653	0.40129
	QL	**0.01312**	**0.01040**	**1.45997**	**0.13109**	**0.42203**

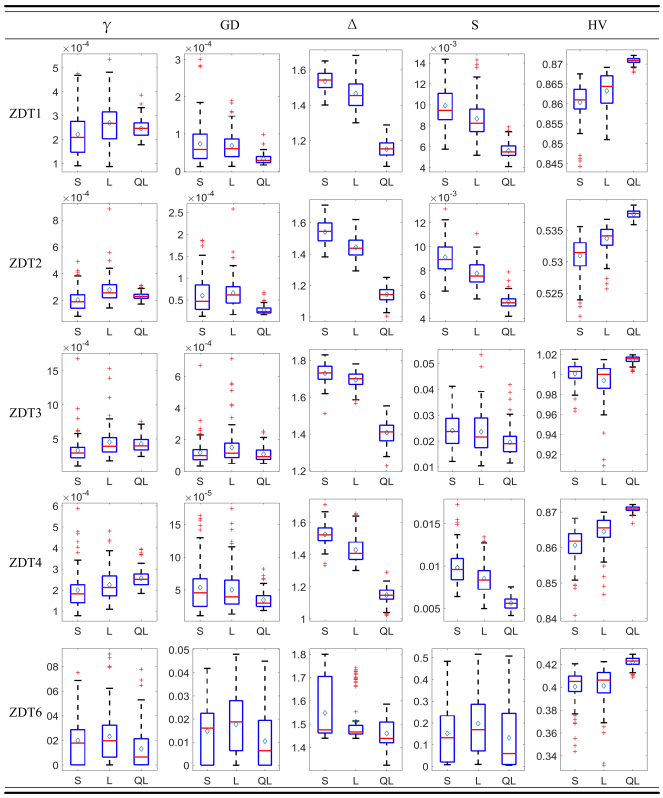


Fig. 9. The boxplots for solving multi-objective optimization problems.

Through the selection of state and action, the design of Q table and the determination of reward function, we have finally realized the online adaptive parameter control of PSO algorithm with the help of RL. The statistical results of different experiments illustrate the much better performance of QLPSO when solving the single-objective and multi-objective optimization problems. Future work includes transplantation of the proposed parameter control method to other evolutionary algorithms, comparison with other parameter control strategies and application in more complex real-world situations.

REFERENCES

[1] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in IEEE International Conference on Evolutionary Computation Proceedings, pp. 69-73, 1998.

[2] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," Computers & Operations Research, vol. 33, pp. 859-871, 2006.

[3] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," IEEE Transactions on Evolutionary Computation, vol. 8, pp. 240-255, 2004.

[4] S. F. Li and C. Y. Cheng, "Particle swarm optimization with fitness adjustment parameters," Computers & Industrial Engineering, vol. 113, pp. 831-841, 2017.

[5] A. E. Eiben, M. C. Schut, and A. R. D. Wilde, "Is Self-adaptation of Selection Pressure and Population Size Possible?-A Case Study," in International Conference on Parallel Problem Solving From Nature, pp. 900-909, 2006.

[6] A. Aleti and I. Moser, "A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms," ACM Computing Surveys, vol. 49, pp. 1-35, 2016.

[7] N. Iwasaki, K. Yasuda, and G. Ueno, "Dynamic parameter tuning of particle swarm optimization," IEEE Transactions on Electrical & Electronic Engineering, vol. 1, pp. 353-363, 2010.

[8] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," Applied Mathematics & Computation, vol. 219, pp. 4560-4569, 2013.

[9] S. W. Leung, S. Y. Yuen, and K. C. Chi, "Parameter control system of evolutionary algorithm that is aided by the entire search history," Applied Soft Computing, vol. 12, pp. 3063-3078, 2012.

[10] G. S. Piperagkas, G. K. Georgoulas, K. E. Parsopoulos, C. D. Stylios, and A. Likas, "Integrating particle swarm optimization with reinforcement learning in noisy problems," in Conference on Genetic & Evolutionary Computation, 2012.

[11] Y. Z. Hsieh and M. C. Su, "A Q -learning-based swarm optimization algorithm for economic dispatch problem," Neural Computing & Applications, vol. 27, pp. 2333-2350, 2015.

[12] J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," Computers & Industrial Engineering, vol. 110, pp. 75-82, 2017.

[13] J. Kennedy and R. Eberhart, "Particle swarm Optimization," in IEEE International Conference on Neural Networks, vol.4, pp. 1942-1948, 2002.

[14] C. A. Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization," in Congress on Evolutionary Computation, pp. 1051-1056, 2002.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 182-197, 2002.

[16] Q. Wang and Z. L. Zhan, "Reinforcement learning model, algorithms and its application," in International Conference on Mechatronic Science, Electric Engineering and Computer, pp. 1143-1146, 2011.

[17] B. J. A. Kröse, "Learning from delayed rewards," Robotics and Autonomous Systems, vol. 15, pp. 233-235, 1995.

[18] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in Congress on Evolutionary Computation, 2002.

[19] N. Riquelme, C. V. Lücken, and B. Baran, "Performance metrics in multi-objective optimization," in Latin American Computing Conference, pp. 1-11, 2015.