IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 14, 2021, accepted April 4, 2021, date of publication April 9, 2021, date of current version April 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072199

Research on Large-Scale Bi-Level Particle Swarm Optimization Algorithm

JIA-JIA JIANG, WEN-XUE WEI, WAN-LU SHAO, YU-FENG LIANG, AND YUAN-YUAN QU
College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
Corresponding author: Wen-Xue Wei (wwxjyh@163.com)

ABSTRACT Targeting at the slow convergence and the local optimum problems of particle swarm optimization (PSO), a large-scale bi-level particle swarm optimization algorithm is proposed in this paper, which enlarges the particle swarm scale and enhances the initial population diversity on the basis of multi-particle swarms. On the other hand, this algorithm also improves the running efficiency of the particle swarms by the structural advantages of bi-level particle swarms, for which, the upper-level particle swarm provides decision-making information while the lower level working particle swarms run at the same time, enhancing the operation efficiency of particle swarms. The two levels of particle swarms collaborate and work well with each other. In order to prevent population precocity and slow convergence in the later stage, an accelerated factor based on increasing exponential function is applied at the same time to control the coupling among particle swarms. And the simulation results show that the large-scale bi-level particle swarm optimization algorithm is featured in better superiority and stability.

INDEX TERMS Bi-level particle swarm, swarm intelligence, particle swarm optimization, large-scale particle swarm.

I. INTRODUCTION

Particle Swarm Optimization (PSO) is an evolutionary computing technique [1], [2], which is derived from the study on birds predation behavior. Similar to the genetic algorithm, PSO is also an iteration-based optimization tool. It firstly initializes a set of random solutions in the system, takes each individual as the particle having no weight or volume in n-dimension space, and then searches the optimum value by iteration, so that the particles in the solution space could follow the optimum particle to search. Due to its advantages of rapid searching speed, good initial convergence, and easy realization, the PSO algorithm has been widely used in many fields [3]–[11]. However, it also has disadvantages, especially when solving complicated problems, such as bad diversity performance in later stage, reduced evolution speed, and unsatisfactory optimization accuracy, etc. Though PSO has little parameters to be adjusted, improper setting of parameters could make the algorithm trapped in problems like ''precocity'' and local optimum, etc.

The associate editor coordinating the review of this manuscript and approving it for publication was Seyedali Mirjalili.

To solve these problems, many scholars have conducted researches in various aspects. Some tried to improve the algorithm by parameter setting, for which, the inertia weight [12]–[15] and the accelerated factor [16], [17] were mainly adjusted. That's because the inertial weight setting affects both the global and the local search capabilities of the algorithm. The appropriate inertial weight should be selected to balance the search capability and algorithm accuracy. On the other hand, the setting of learning factor determines the influences of individual historical optimal value and group historical optimal value on particle moving trajectory. Too large learning factor may make the particles jump out of the optimal area. But too small learning factor may cause particle oscillation in areas far away from the target. In addition, some other scholars tried to improve the optimization performance of the algorithm by changing population topology structure [18] because the change of topology structure can avoid population diversity loss and prevent the algorithm from being trapped in local optimum problem. However, there have no topology structure that is suitable for all benchmark functions. The selection of topology structure is actually related to specific problem model. Besides, the way to form mixed PSO [19] by combining with genetic algorithm [20]–[22], ant colony algorithm [23]–[25], or other optimization algorithms,

can make up the PSO defects to some certain extent. But all these algorithm improvement are based on single-swarm, which though improve the algorithm performance to certain extent, still have their problems like precocity convergence and low solution accuracy. With the in-depth study on algorithm, some scholars started to study multi-swarms in order to improve the algorithm. Luo Dexiang divided a single particle swarm into three particle swarms, and made comparison on the optimal values obtained by the independent operation of the three sub-swarms in order to get the global optimum [26]. Although this algorithm prevent the local optimum problem to a certain extent, it sacrifices the accuracy. In literature [27], Lovbjerg *et al.* divided the population into multiple sub-populations. And in this paper, the optimal particles of the sub-populations was applied to replace the population optimal particles in the basic PSO algorithm speed update formula, which reduces the local optimum risk of algorithm. However, the only interaction among sub-populations is the parental reproduction among different sub-populations, and the information exchange between sub-populations is insufficient. Liu *et al.* [28] used the K-means clustering algorithm to divide the population into several sub-populations and strengthen the information exchange between particles by periodically reconstructing the sub-populations. In order to avoid local optimum, some scholars proposed a particle swarm optimizer with two differential mutation [29], and some others introduced the crossover operator to improve the performance of the bl-level particle swarm optimization [30]. Although this algorithm prevents the local optimum problem to some certain extent, it prolongs the running time of the algorithm.

Targeting at the above problems, we propose a new large-scale bi-level particle swarm optimization algorithm, which can better balance the global search and local search ability of the algorithm and maintain good calculation accuracy while reducing the running time; in the bi-level particle swarm optimization structure, the learning factor strategy of exponential function distribution is proposed to guide the lower-level working particle swarm optimization, the upper-level particle swarm is used to optimize the evolution direction while the lower-level particle swarm is used to increase population diversity, and the upper and lower levels operate in coordination and work together; the scale of the particle swarm is enlarged and the lower level particle swarm operates in parallel, which improves the population diversity of the population in the initial stage, improves the accuracy of the algorithm and reduces the running time. Finally, we verify the effectiveness of the algorithm through comparative experiments.

The rest part of this paper is composed in structure as below: in Chapter 2, we briefly review the relative methods of basic particle swarm optimization (BPSO); in Chapter 3, we introduce the method proposed in this paper in detail; and in Chapter 4, we introduce our experiment in detail and the final experimental results; and finally, we provide a summary of our research work and a prospect for future work in Chapter 5.

II. THE BASIC PARTICLE SWARM OPTIMIZATION ALGORITHM THEORY

As for the basic particle swarm optimization it firstly initializes n random particles as a particle swarm $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$, and each particle refers to one potential optimal solution, d refers the dimensions which equals the number of unknowns of to-be-solved problems. Suppose that the position of No.i particle in d-dimension search space is expressed as: $\boldsymbol{X}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T$, and the velocity is expressed as $\boldsymbol{V}_i = [v_{i1}, v_{i2}, \ldots, v_{id}]^T$, all particles have a corresponding fitness value that is determined by the optimized function, and the fitness value is used to judge the particle performance. During each iteration, each particle should be recorded for two positions: one is $\boldsymbol{P}_i = [P_{i1}, P_{i2}, \ldots, P_{id}]^T$, which is the best position that Particle X_i has searched, and the other is $\boldsymbol{G} = [G_1, G_2, \ldots, G_d]^T$, which is the best position that the particle swarm has searched. Each particle adjusts its updated position by tracking these two positions. And the PSO achieves optimum searching by such population circulated iteration. For the update formula, please formula (1) and (2).

$$V_{id}(k+1) = \omega V_{id}(k) + c_1 r_1 (P_{id}(k) - X_{id}(k)) + c_2 r_2 (G_d(k) - X_{id}(k)) \quad (1)$$

$$X_{id}(k+1) = X_{id}(k) + V_{id}(k+1) \quad (2)$$

In formula (1) and (2), ω is the inertia weight, which significantly affects the search capability of the algorithm. Larger ω is good for the global search while the smaller one is good for the local search; k refers to the current iteration times; c_1 and c_2 refer to the accelerated factors, which are used to adjust the step length of the particle moving towards the individual best position and the global best position. But too large or too small accelerated factors may make the particles far away from the optimum or result in particle oscillation. r_1 and r_2 are the random numbers distributed between [0,1]. The pre-determined maximum iteration times or the optimization result satisfying the accuracy requirement should be the condition for iteration termination of the basic particle swarm optimization. The global extreme value $\boldsymbol{G}$ at the time when the algorithm terminates is the final optimum. In order to improve the search efficiency and prevent particles from getting out of the search space, particle velocity at each dimension is restricted between $[-\mathrm{V}_{\max}, \mathrm{V}_{\max}]$. And commonly, $\mathrm{V}_{\max}$ will not exceed the width of the particle. In the same way, the particle position at each dimension is also restricted between $[-\mathrm{X}_{\max}, \mathrm{X}_{\max}]$. If there's particle leaving the solution space, this range would be applied to re-update particle information.

The BPSO is not sensitive to environment changes, and can be easily affected by $\boldsymbol{P}_i$ or $\boldsymbol{G}$, so that it is difficult to converge to the global optimum. Especially in the complex multi-modal functions in high-dimensional space, these functions have many local optimums, which can easily attract particle swarms, making the algorithm fall into the local optimum and precocity convergence. Experiments indicate that the ways to

change the inertia weight ω or accelerated factors c_1 and c_2 have no significant effects in improving the efficiency and stability of the algorithm.

III. LARGE-SCALE BI-LEVEL PARTICLE SWARM OPTIMIZATION ALGORITHM

A. BI-LEVEL MULTI-PARTICLE SWARM

Most optimization made to particle swarms previously were targeted at current swarm evolution. However, PSO for single swarm particles could fall into local optimum easily, and has disadvantages of low search efficiency and bad stability performance in later stage. Compared to the PSO for single swarm of particles, multi-particle swarm optimization algorithm is featured in a significant structural advantage: when a specific particle swarm falls into local optimum, other particle swarms can provide it with their particle information to help it get rid of local optimum. Information exchange among multiple particle swarms can greatly improve the search efficiency of particle swarms in later stage. On the other hand, it is generally believed that the larger the population size, the higher accuracy and the better stability of the PSO. In order to get the complicated problem solved faster, the particle swarm size can be enlarged to broaden the search range of the particle swarm and enhance the exploration capability of the population.

To balance the exploration and development capabilities of particle swarms, a large-scale bi-level PSO was proposed in this paper, which makes the particle swarms play a greater role with large scale, and overcome the local optimum and slow convergence of single-level particle swarm by introducing bi-level particle swarm structure. For the algorithm structural diagram, please see Figure 1 below.

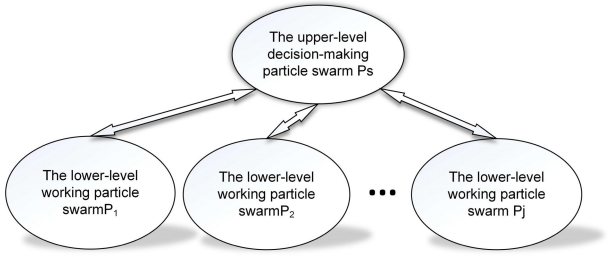


FIGURE 1. Schematic diagram of large-scale bi-level particle swarm structure.

In Figure 1, the large-scale bi-level PSO algorithm is divided into upper and lower levels. The lower level particles, which contain rich information, are divided into small sub-populations. Formula (3) shows the velocity information of No.i particle in No.j sub-population of the lower level. Formula (4) shows the current position of No.i particle in No.j sub-population of the lower level. And Formula (5) shows the individual extreme value of No.j sub-population of the lower level.

$$\boldsymbol{V}_{jid} = \left[V_{ji1}, V_{ji2}, \ldots, V_{jiD}\right]^T \tag{3}$$

$$\boldsymbol{X}_{jid} = \left[X_{ji1}, X_{ji2}, \ldots, X_{jiD}\right]^T \tag{4}$$

$$\boldsymbol{P}_{jid} = \left[P_{ji1}, P_{ji2}, \ldots, P_{jiD}\right]^T \tag{5}$$

Each lower-level working particle swarm can find an optimal solution of sub-population, which affects the movement of particles in the swarm to a certain extent. The lower-level particle swarm provides optimal particle information, and uses the mean value of the optimal particle information to seek the global optimum. And the upper-level decision-making particle swarm $\boldsymbol{P_s}$ collects the optimal information of the lower-level working particle swarm $\boldsymbol{P}_j$ (j is the label of the lower-level particle swarm), and processes the optimal information to generate decision information and feed it back to each lower-level working particle swarm. In formula (6), the population with size of N can be divided into m sub-populations. The size of sub-population $\boldsymbol{P}_j$: $\left|P_j\right| = \text{floor}\,(\text{N/m})$, where floor is rounded down, and $\text{j} = 1, 2, \ldots, \text{m}$. In formula (7), $\boldsymbol{X}_{xs}$ is used as the decision-making information of the upper-level particle swarm to guide the evolution direction of the lower-level particle swarms, conduct the lower-level working particle swarms to explore the area that may possibly contain the optimum, and make the lower-level particles fly toward the optimal position. When a certain particle swarm in the lower-level falls into the local optimum, the upper decision-making particle swarm can help it get rid of the local optimum.

$$\bar{\boldsymbol{P}} = \frac{\boldsymbol{P_{1g}} + \boldsymbol{P_{2g}} + \ldots + \boldsymbol{P_{mg}}}{m} \tag{6}$$

$$\boldsymbol{X}_{xs} = \bar{\boldsymbol{P}} - \boldsymbol{X}_{jid} \tag{7}$$

In the early stage of the iteration, the lower-level working particle swarms are distributed and scattered randomly. In order to enhance the exploration capability of the algorithm in the early stage, the guidance of the upper-level decision-making particle swarm to the lower-level working particle swarms should be loosely coupled. In the later stage when the lower-level particle swarms tend to the optimal solution and the lower-level working particle swarms have richer information, the way to strengthen the guidance effects of the upper-level decision-making particle swarms on the lower-level working particle swarms can better improve the exploration and development capabilities of the algorithm. In this case, it is proper to adopt the exponential distribution learning factor strategy.

$$R = x^{\alpha}, x = \frac{i}{s}\,(\alpha \geq 1) \tag{8}$$

$$v_{jid}^{k+1} = \omega^{v_{jid}^{k}+c_1 r_1}\left(P_{jid}^{k} - X_{jid}^{k}\right) + Rc_3 X_{xs} \tag{9}$$

$$X_{jid}^{k+1} = X_{jid}^{k} + V_{jid}^{k+1} \tag{10}$$

The numerical value of a in formula (8) needs to be set according to the specific processing cases. Combining formulas (7) and (8), the iterative formulas evolve from formulas (1) and (2) to formulas (9) and (10).

B. ANALYSIS ON PARAMETERS OF POPULATION SIZE

As for the BPSO, the population size N is an important parameter, representing the number of particles in the particle

swarm and playing a significant role in algorithm convergence speed, accuracy and stability. Generally, the number of particles in the research process is determined by the complexity of the problem. In this paper, N = 50, 200, 400, 800 were selected to analyze its effect on the algorithm performance. Figure 2 indicates the curve of fitness value changes with the increase of iteration times when N is taken with different numerical values.

In the experiment, the number of iterations was set as 500, and the population size N was changed. Then, 30 times of evolutionary operations were conducted on the testing functions respectively. Then, the effects of N on algorithm performance are as shown in Figure 2. It can be seen from Figure 2 that, with the increase of population size, the PSO evolutionary operation shows better effects: the greater the particle population size, the higher accuracy the search. But for small number of particles, the exploration on particle swarm seems inadequate. Especially for complicated and high-dimension problems, large scale population always shows more significant advantages.

But when the number of particle swarms reached 200, 400, and 800, though the mean optimal fitness value of the particle swarm was improved compared to cases with N below 100, the improvement effect was not so significant with the increase of particle number, and the optimal fitness values varied little. And in single particle swarm, the great the population size, the faster the convergence velocity at the initial stage of iteration. Moreover, during the aforesaid convergence process, it lost diversity easily and failed to well maintain the global convergence capability of the algorithm. With the enlargement of population size, the computation overhead increased exponentially, and the running time also increased. The PSO for single particle swarm showed no obvious advantages in evolutionary operation, and brought no more gains to the algorithm. However, these problems could be well solved in bi-level particle swarm optimization algorithm because it can exert its advantages of large scale population and structure to achieve balance of PSO between accuracy and stability improvement, and running time reduction.

C. PARALLEL RUNNING OF MULTI-PARTICLE SWARMS

For complex problem processing, the search space of multi-particle swarm evolutionary calculation enlarges sharply. And it usually costs a long time to operate on a single CPU due to the low operation efficiency. In this case, the parallel computation could be introduced among multiple particle swarms to effectively resolve the overlong computation time problem for large-scale computation by collaborating parallel work of several CPUs.

The number of working particle swarms in lower-level should be firstly determined, and then the number of sub-threads in the primary thread should be set according to the number of working particle swarms. After that, each thread shall be assigned with specific tasks. There are two ways of information exchanges: synchronous and asynchronous.

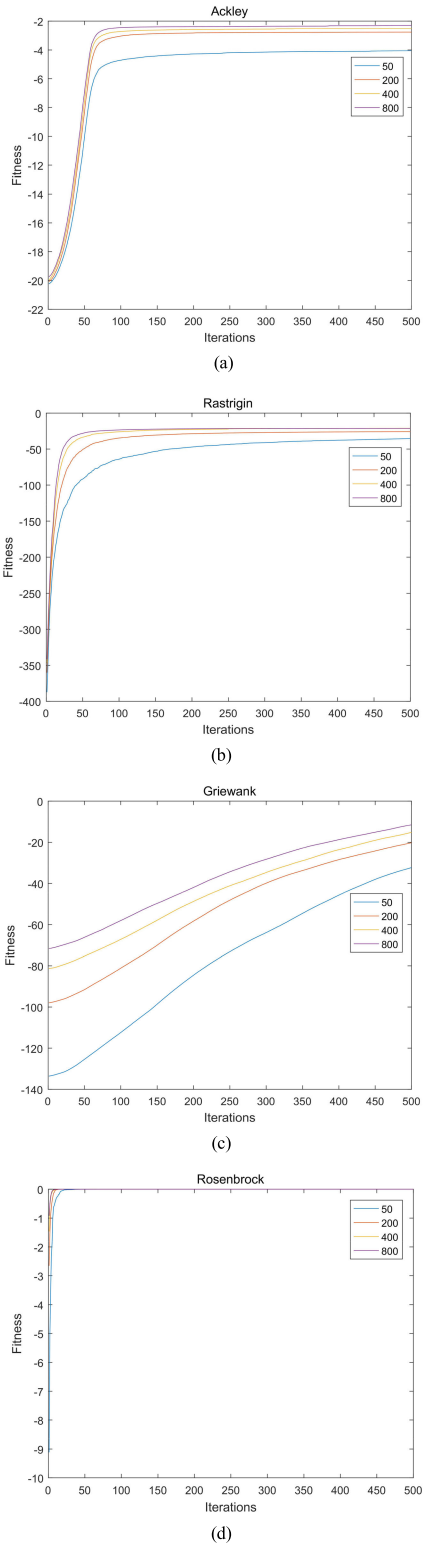


FIGURE 2. Schematic diagram of convergence process under different population sizes. (a) Ackley function; (b) Rastrigin function; (c) Griewank function; (d) Rosenbrock function.

Asynchronous information exchange refers to: when one thread accomplishes update of individual optimal position and velocity, it will provide the information to upper-level decision-making particle swarm immediately for judging the

flying direction next time. And the synchronous information exchange refers to: particles of all threads accomplish update of position and velocity first, and then evaluate and update the global optimum individual position and velocity based on these information in order to make judgment on the flying direction next time. As for this paper, the parallel algorithm was adopted, which is essentially the simulation to the bird flock predatory behavior. It simulates the behavior that the bird leader divides the bird flock into multiple groups. After a time of search, each group informs the bird leader about the best position they find. And the leader judges the optimal position that is most likely to find food, and inform the bird flock to forge ahead toward the direction. Therefore, this algorithm adopts synchronous mode in parallel information exchange: when particles of all threads accomplish the updates, the upper-level decision-making particle swarm collects the optimum particle information of the working particle swarms of the lower-level, and then process information to generate decision-making information, and inform the particle swarms in lower-level. Therefore, besides the effects of its own optimal position and optimal particle position of lower-level working sub-populations, each particle of the swarm is also affected by the overall decision-making information of the whole group, thus to update the flying position information next time.

During the evolutionary operation process, the multi-particle swarm optimization algorithm divides the particle swarm into several sub-particle swarms, which are evenly distributed in the solution space to ensure the diversity of particles. Each particle swarm only needs to accomplish its own internal evolution. At the end of each iteration, particle swarms conduct information communication, so that when any sub-particle swarm falls into local optimum, other particle swarms will provide their position information to it. These information provides direction for the swarm trapped in local optimum, and helps it get rid of the local optimum problem. Please see Figures 3-4.

Large-scale bi-level particle swarm optimization algorithm can well solve the problem that the particle swarms fall into local optimum, and help those sub-particle swarms trapped in local optimum escape from the local optimum during the iterative process. The lower-level working particle swarms mainly have three types of status, first: all the lower-level working particle swarms tend to the global optimum. And at this time the upper-level decision-making particle swarm can provide better guidance by integrating the status of each lower-level working particle swarm; the second type: some lower-level working particle swarms fall into local optimum while the rest part tends to the global optimum, then the upper-level decision-making particle swarm will guide the lower-level working particle swarm through comprehensive information, directing the particle swarms trapped in the local optimum to the global optimum and helping them escape from local optimum; the third type: all lower-level working particle swarms tend to local optimum: suppose that the probability of lower-level working particle swarms falling into the local optimum is x%, then the probability of all lower-level working particle swarms falling into the local optimum would be $(x\%)^m$(m refers to the number of working particle swarms in the lower level). Compared with a single particle swarm, the probability that all working particle swarms in a large-scale bi-level particle swarms falling into a local optimum is greatly reduced.

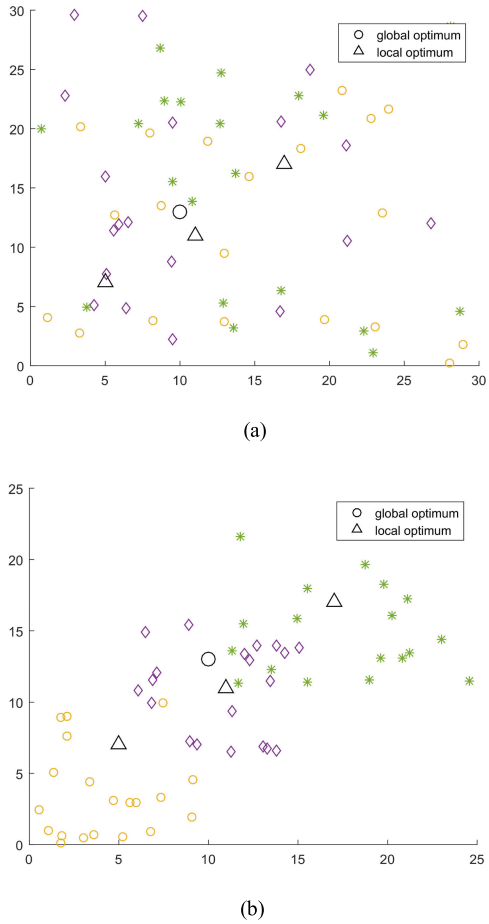


FIGURE 3. Schematic diagram of falling into local optimum. (a) Initial status of lower-level working particle swarms of bi-level PSO;(b) certain lower-level working particle swarm of bi-level PSO falls into local optimum.

D. ALGORITHM STEPS

Step 1: Construct a bi-level particle swarm structure.

(1) Divide the global particle swarm into several lower-level sub-swarms of working particles.

(1) Create an upper-level decision-making particle swarm space.

Step 2: Initialize the lower-level working particle swarms, and initialize parameters, particle velocity, position, and calculate the fitness value and individual extreme value of the particle.

Step 3: Extract the optimal particle information from each lower-level working particle swarm, and provide to the upper-level decision-making particle swarm.

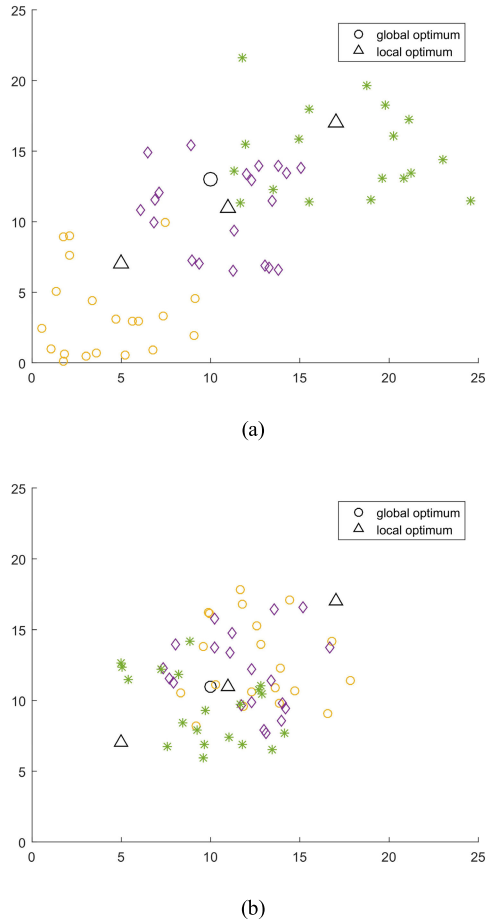


FIGURE 4. Schematic diagram of escaping from local optimum. (a) Certain lower-level working particle swarm of bi-level PSO falls into local optimum;(b) lower-level working particle swarm of bi-level PSO escape from local optimum.

Step 4: The upper-level decision-making particle swarm guides the evolutionary direction for the lower-level working particle swarms.

(1) The upper-level decision-making particle swarm calculates the decision-making information.

(2) Provide the decision information to the lower-level working particle swarms according to the learning factor distributed by the exponential function, and conduct the updates of velocity and position.

Step 5: The lower-level working particle swarms run in parallel and calculate the fitness values.

Step 6: Judge if the algorithm meets the termination condition (the algorithm has reached the required times of iterations), if it meets the requirements, go to step 7, otherwise go to step 3.

Step 7: Output the global optimum, and the algorithm ends.

IV. SIMULATION ANALYSIS

A. PARAMETER SETTING

In this section, the large-scale bi-level particle swarm optimization method is evaluated from various aspects by an array of experiments conducted in benchmark functions.

TABLE 1. The detailed settings.

Name	Detailed settings
Hardware	
CPU	Inter(R) Core(TM) i5-6200U
Frequency	2.4GHz
RAM	8.00GB
Hard drive	466GB
Software	
Operating system	Windows 10
Language	MATLAB R2016a(9.0)

In order to get an unbiased comparison of CPU times, all the experiments were performed on a same PC, which was configured with detailed settings as shown in Table 1.

Eleven different benchmark functions were used to evaluate the large-scale bi-level particle swarm optimization algorithm that we proposed. The test functions are divided into two groups: the unimodal and the multi-modal. The unimodal functions (F1-F5) are suitable for benchmarking the exploitation of algorithms since they have one global optimum and no local optima. On the contrary, the multi-modal functions (F6-F11) have a massive number of local optima and are helpful for examining the exploration and local optima avoidance of algorithms. The expressions and properties of these benchmarks are presented in Table 2 and Table 3 respectively. Parameters of experiment test are as listed in Table 4.

B. COMPARISONS OF THE LARGE-SCALE BI-LEVEL PARTICLE SWARM OPTIMIZATION ALGORITHMS WITH OTHER METHODS

In terms of the benchmark problems, the performance of the large-scale bi-level particle swarm optimization algorithms was compared with six other optimization algorithms. The methods included in the comparative study are BPSO, adaptive particle swarm optimization(APSO) [31], elephant herding optimization(EHO) [32], moth-flame optimization algorithm (MFO) [33], monarch butterfly optimization (MBO) [34] and earthworm optimization algorithm (EWA) [35].

Tables 5 and 6 record the average and the best results of 30 tests respectively. Table 5 shows that, on average, the algorithm in this paper outperforms the effects of other methods on seven of the eleven benchmarks (F1, F2, F6-F8, F10 and F11) when searching for the minimum value of the function. EHO and MFO are the second most effective methods and show the best performance on eleven benchmarks (F5, F9, and F3, F4, respectively). It can be seen from Table 6 that, in five of the eleven benchmarks (F1, F2, F7, F10, and F11), the algorithm in this paper is better than other methods. EHO also shows better performance than other algorithms in the five benchmarks (F4-F6, F8, and F9), while the MFO ranking the third most efficient and showing the best performing on the benchmarks F3 when multiple runs are made. It can be seen that the algorithm in this paper can greatly improve the performance of the algorithm.

Furthermore, the optimizing processes of all algorithms are given in Figure 5-15. The values shown in these figures are

TABLE 2. Benchmark functions.

No.	Name	Definition
F1	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$
F2	Schwefel 2.22	$f(\vec{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F3	Schwefel 1.2	$f(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$
F4	Schwefel 2.21	$f(\vec{x}) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$
F5	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
F6	Quartic with noise	$f(\vec{x}) = \sum_{i=1}^{n} \left(i \cdot x_i^4 + random(0,1) \right)$
F7	Rastrigin	$f(\vec{x}) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$
F8	Ackley	$f(\vec{x}) = -20e^{\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}\right)} - e^{\left(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)\right)} + 20 + e$
F9	Griewank	$f(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \Pi_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
F10	Penalty #1	$f(\vec{x}) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$
F11	Penalty #2	$f(\vec{x}) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$

TABLE 3. Properties of benchmark functions, lb denotes lower bound, ub denotes upper bound, opt denotes optimum point.

No.	Function	lb	ub	opt	d	Modality
F1	Sphere	-5.12	5.12	0	30	Unimodal
F2	Schwefel 2.22	-10	10	0	10	Unimodal
F3	Schwefel 1.2	-100	100	0	10	Unimodal
F4	Schwefel 2.21	-100	100	0	10	Unimodal
F5	Rosenbrock	-30	30	0	30	Unimodal
F6	Quartic with noise	-1.28	1.28	0	10	Multimodal
F7	Rastrigin	-5.12	5.12	0	30	Multimodal
F8	Ackley	-32	30	0	30	Multimodal
F9	Griewank	-600	600	0	10	Multimodal
F10	Penalty #1	-50	50	0	10	Multimodal
F11	Penalty #2	-50	50	0	10	Multimodal

the optimal function optima achieved from 30 runs. Here, all the values are true function values without being normalized.

Figure 5 shows the value of F1 function obtained by seven methods. Value in the figure is the function value of the spherical function of F1, also known as De Jong's function with global value of $F1_{min} = 0$, so it is easy to solve. It is known from FIGURE 5 that, the large-scale bi-level particle swarm optimization algorithm has the fastest convergence rate towards the global solution, which is better than all other methods.

TABLE 4. Parameters setting for experiment.

parameter name	size
population size	800
c_1,c_2	1.49445
c_3	6.49445
α	1.2
iterations	500
algorithm run times	30

Figure 6 reveals the function values for F2 Schwefel 2.22 function. Viewing from Figure 6, although our algorithm converges slowly in the early stage, it finally

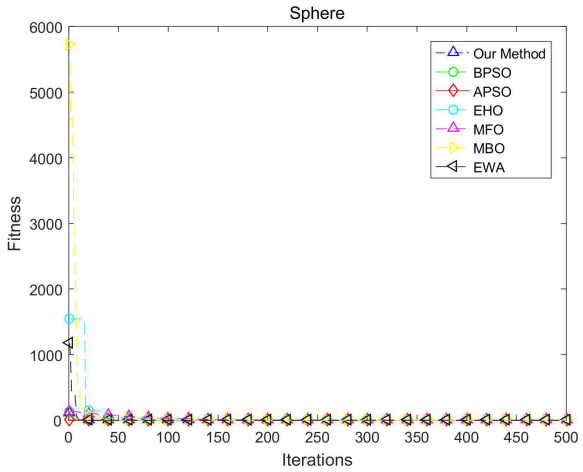


FIGURE 5. Performance comparison on the F1 sphere function.

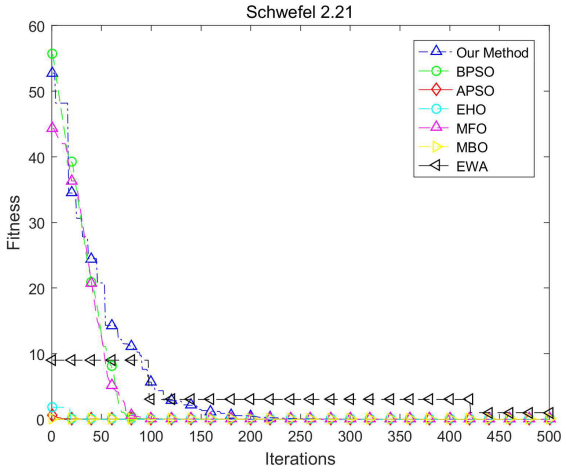


FIGURE 8. Performance comparison on the F4 schwefel 2.21 function.

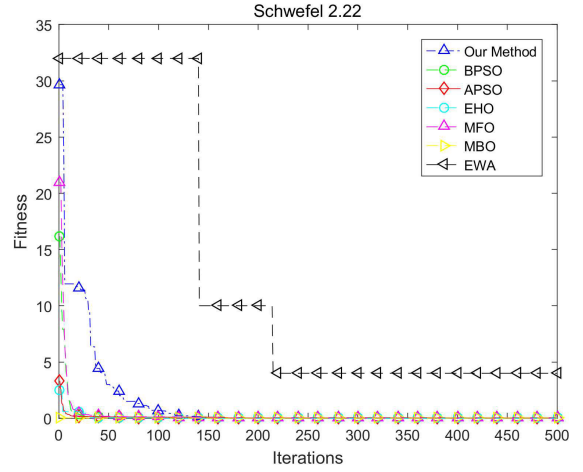


FIGURE 6. Performance comparison on the F2 schwefel 2.22 function.

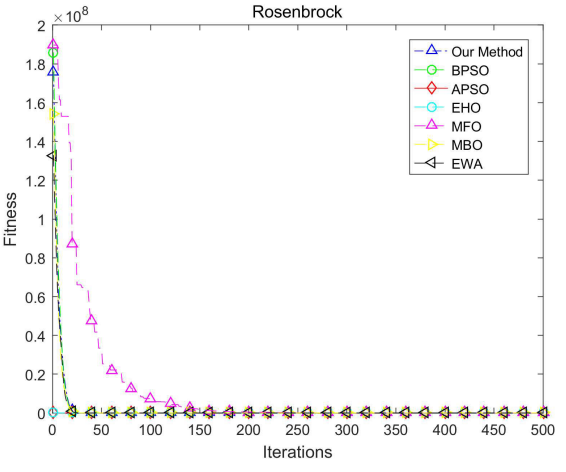


FIGURE 9. Performance comparison on the F5 rosenbrock function.

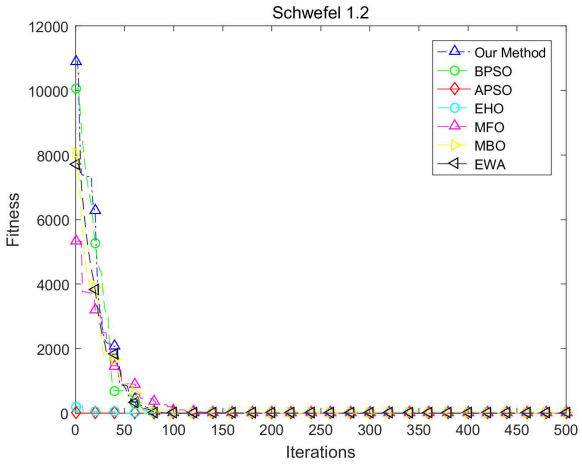


FIGURE 7. Performance comparison on the F3 schwefel 1.2 function.

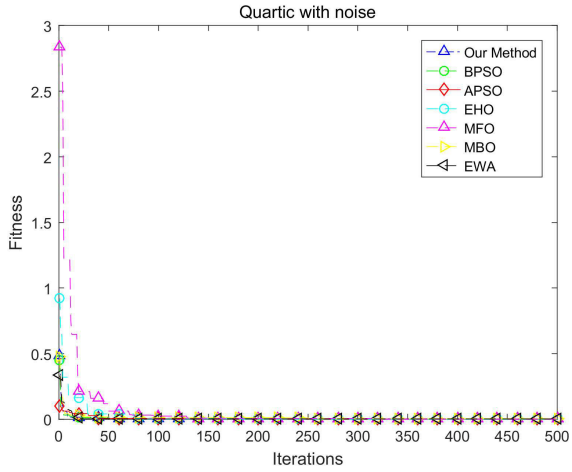


FIGURE 10. Performance comparison on the F6 quartic with noise function.

shows the optimal performance in this unimodal benchmark function.

Figure 7 reveals the function values for F3 Schwefel 1.2 function. As for the unimodal function in Figure 7, MFO algorithm performs better than the other six algorithms, EHO also shows a good convergence speed, and large-scale bi-level particle swarm optimization algorithm ranks the third.

Figure 8 reveals the function values for F4 Schwefel 2.21 function. It can be known from Figure 8 that, both MFO and EHO algorithms perform well in finding the global minimum,

TABLE 5. Mean fitness function values.

	MY	BPSO	APSO	EHO	MFO	MBO	EWA
F1	0.0002	0.0005	0.0099	1.678	1.7498	7.6274	0.0326
F2	1.69E-09	0.0115	0.0766	0.0025	0.0134	0.0168	2.6263
F3	1.47E-05	0.0001	0.0034	0.0527	3.48E-06	0.0004	0.0002
F4	0.0034	0.0034	0.0214	0.0011	1.93E-05	0.0437	1.4513
F5	29.0852	47.5298	32.5200	1.0479	7193.23	171.2913	87.8424
F6	0.0012	0.0014	0.0057	0.0021	0.0015	0.0041	0.0022
F7	0.0009	23.6475	8.8835	0.0152	82.30	18.014	14.0007
F8	0.0023	1.6815	0.2173	0.0032	0.8797	5.6586	3.8376
F9	0.1461	0.3116	33.2353	0.0003	0.1214	0.5973	0.407
F10	3.04E-16	5.88E-06	6.52E-05	0.0037	1.39E-05	8.10E-06	3.73E-06
F11	1.48E-15	2.08E-05	0.0003	0.0080	6.53E-05	4.92E-06	1.57E-05

TABLE 6. Best fitness function values.

	MY	BPSO	APSO	EHO	MFO	MBO	EWA
F1	1.24E-05	0.0002	0.0068	1.25E-05	0.0008	0.0405	0.0016
F2	2.18E-10	0.0063	0.0372	1.23E-05	0.0051	0.0017	2
F3	1.39E-05	2.94E-05	0.0008	4.07E-08	1.05E-08	0.00008	0.00007
F4	0.0012	0.0013	0.0057	1.15E-06	3.95E-06	0.0011	1
F5	28.2832	24.8569	29.9446	9.96E-06	83.4063	18.7355	43.9682
F6	0.00016	0.00022	0.00166	3.12E-03	0.00034	0.00015	0.00013
F7	0.00019	11.1065	1.9793	0.0013	35.84	8.8756	5.3236
F8	0.0010	0.0669	0.1605	1.90E-06	0.1323	5.0588	3.3694
F9	0.0129	0.1021	9.0513	0.0001	0.0320	0.1259	0.2142
F10	7.38E-18	2.07E-06	2.23E-05	4.17E-08	3.48E-06	4.18E-06	2.49E-06
F11	1.09E-18	4.92E-06	7.09E-05	7.99E-07	1.26E-05	3.44E-05	0.78E-05

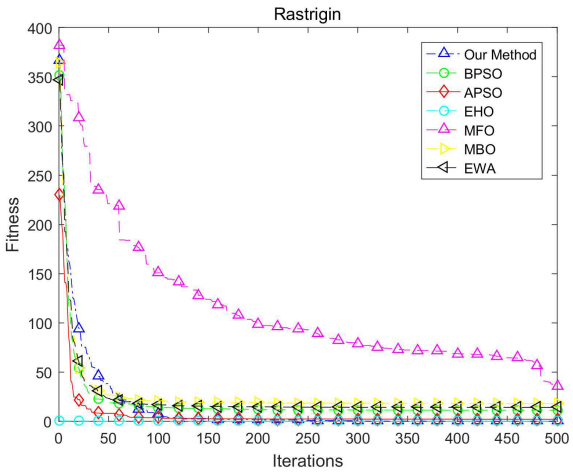


FIGURE 11. Performance comparison on the F7 Rastrigin function.

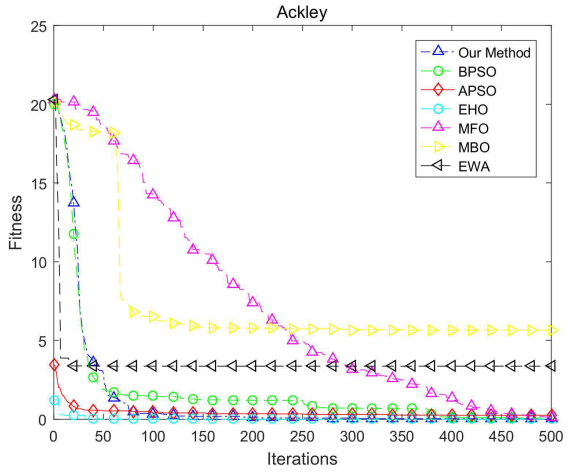


FIGURE 12. Performance comparison on the F8 ackley function.

while other algorithms perform poorly in this benchmark function.

Figure 9 reveals the function values for F5 Rosenbrock function, showing that, the EHO has the fastest convergence speed and the best performance in this benchmark function.

Figure 10 illustrates the values achieved for the seven methods when using F6. In the convergence graph of the optimal value, the EHO algorithm shows extremely fast convergence speed, and at the same time, the algorithm proposed in this paper also performs well in this benchmark function, providing good results and showing great advantages, but the MFO algorithm shows the slowest convergence speed.

Figure 11 reveals the function value of the F7 Rastrigin function, which is a complex multimodal function with a unique global minimum of $F7_{\min} = 0$ and several local optimals. For solving F07, the method may converge to a local value. Therefore, a method that can keep a larger diversity is more likely to produce better values. The algorithm proposed in this paper has the best performance. In addition, it can be seen from the figure that the MFO algorithm may be trapped in the local optimal value.

Figure 12 shows the values obtained by the seven methods on F8 function, which is a multimodal function with a narrow global minimum basin ($F8_{\min} = 0$) and many minor local optima. Although the algorithm in this paper has a slow convergence speed at the early stage, it performs well after 75 iterations, and the initial values of all the methods are almost the same. In the end, the EHO algorithm surpasses the other six methods, and EWA and MBO may fall into local optima.

TABLE 7. The standard deviation of different methods.

	MY	BPSO	APSO	EHO	MFO	MBO	EWA
F1	0.00013	0.00017	0.00217	5.1716	6.6508	15.4536	4.5263
F2	1.10E-09	0.0044	0.0209	0.0032	0.0037	0.0312	1.0826
F3	0.0001	8.31E-05	0.0021	0.2077	6.93E-06	0.0347	0.0039
F4	0.0012	0.0015	0.0071	0.0014	1.44E-05	0.052	0.8482
F5	0.2881	37.7311	1.3842	3.1509	22597.76	33.9865	28.8521
F6	0.0006	0.0008	0.0033	0.0013	0.0006	0.0002	0.0001
F7	0.0794	6.5332	4.8125	0.0002	36.28	5.6452	6.9695
F8	0.0066	0.6569	0.0295	0.0052	2.507	0.9772	0.5503
F9	0.0866	0.1227	7.6953	0.0014	0.0689	-0.0126	0.1855
F10	7.00E-16	2.59E-06	3.32E-05	0.0190	7.25E-06	3.25E-06	2.25E-06
F11	4.12E-15	1.6E-05	0.0001	0.0391	3.67E-05	0.89E-05	0.96E-05

TABLE 8. Comparison on algorithm running time.

	MY	BPSO	APSO	EHO	MFO	MBO	EWA
F1	105.23	339.92	319.03	613.93	197.57	304.85	289.32
F2	154.51	303.18	309.01	638.05	439.67	256.98	305.88
F3	126.65	422.08	417.68	1558.42	278.34	368.56	298.64
F4	109.02	297.99	315.32	632.80	152.13	259.63	365.96
F5	107.75	313.24	333.49	618.87	195.17	365.22	342.12
F6	112.18	317.43	318.55	800.97	183.62	345.21	331.25
F7	110.24	316.05	317.34	633.84	169.95	368.56	324.75
F8	102.89	274.76	282.56	568.24	186.56	342.15	365.22
F9	117.31	297.10	314.02	300.70	137.68	328.98	329.47
F10	144.85	462.68	493.74	1122.44	307.23	544.56	596.32
F11	146.63	444.48	483.12	1127.41	277.83	452.85	608.96

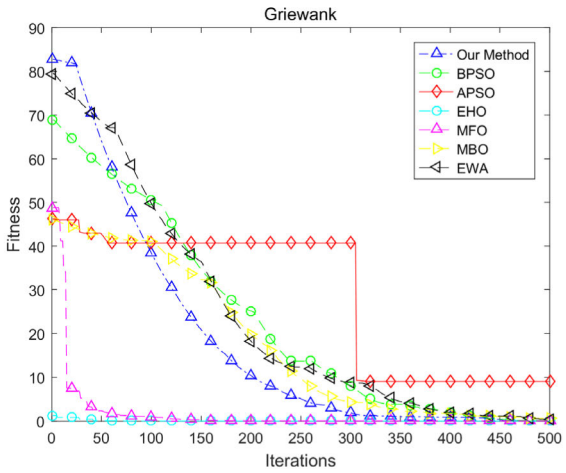


FIGURE 13. Performance comparison on the F9 griewank function.

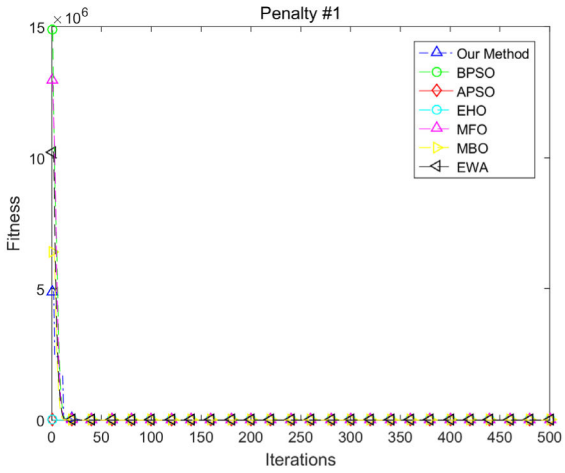


FIGURE 14. Performance comparison on the F10 penalty #1 function.

Figure 13 displays the values for F9 function. The EHO algorithm performs the best in this benchmark function. As the iteration goes on, the convergence speed of the algorithm in this paper gradually surpasses other algorithms and finally ranks the third.

Figure 14 reveals the values for F10 Penalty #1 function. It can be seen from FIGURE 14 that, the large-scale bi-level particle swarm optimization algorithm converges faster in global solution than other algorithms.

Figure 15 shows the values achieved on F11 function. For this benchmark, the large-scale bi-level particle swarm optimization algorithm overtakes all other approaches in the optimization process, which is very similar to F10 in Figure 14.

As shown in Table 8, the running time of the algorithm in this paper reduces greatly compared with those in the other six algorithms, among which, the large-scale bi-level particle swarm optimization algorithm is the most efficient one.

C. COMPARISONS WITH OTHER OPTIMIZATION METHODS BY USING WILCOXON'S RANK-SUM TEST

Based on the final search results of 30 independent trials on every function, we figure out and present the key data in Table 9, which is the p-values of every function of the wilcoxon's rank-sum test with the 5% level of significance between the large-scale bi-level particle swarm optimization algorithm and other optimization methods. Although the large-scale bi-level particle swarm optimization algorithm provides no better results on the test functions (F3, F4, F5 and F9) in Table 5, the p-values in Table 9 show that the results of this algorithm are very competitive.

TABLE 9. P-values of the Wilcoxon rank-sum test over all runs (p >= 0.05 have been underlined).

	BPSO	APSO	EHO	MFO	MBO	EWA
F1	2.03e-09	3.01e-11	3.01e-11	3.01e-11	1.87e-05	3.01e-11
F2	3.01e-11	3.01e-11	3.01e-11	3.01e-11	0.1049	2.62e-11
F3	0.3870	3.01e-11	3.01e-11	3.01e-11	2.62e-11	3.01e-11
F4	3.01e-11	3.01e-11	6.52e-08	3.01e-11	2.62e-11	2.79e-11
F5	0.2225	3.01e-11	3.01e-11	3.01e-11	3.01e-11	2.62e-11
F6	0.4642	6.72e-10	2.15e-06	0.0614	3.01e-11	3.01e-11
F7	3.01e-11	3.01e-11	3.33e-11	3.01e-11	3.01e-11	2.62e-11
F8	3.01e-11	3.01e-11	2.60e-10	3.01e-11	3.01e-11	3.01e-11
F9	1.38e-06	3.01e-11	3.01e-11	0.2458	4.56e-11	4.56e-11
F10	3.01e-11	3.01e-11	3.01e-11	3.01e-11	3.01e-11	3.01e-11
F11	3.01e-11	3.01e-11	3.01e-11	3.01e-11	3.01e-11	2.62e-11

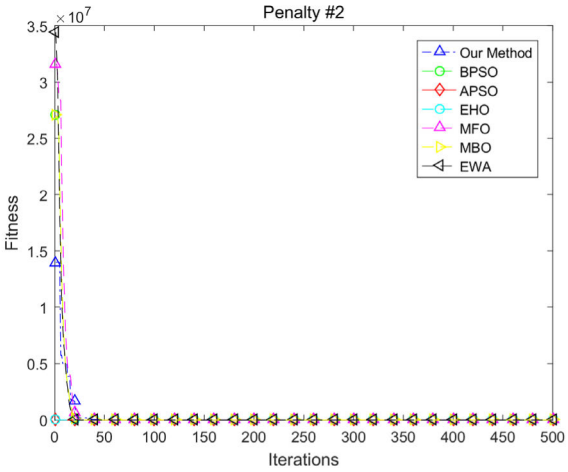


FIGURE 15. Performance comparison on the F11 penalty #2 function.

D. ANALYSIS ON EXPERIMENTAL RESULTS

Compared with other algorithms, although the algorithm proposed in this paper has a slow convergence speed in the early stage, it can hardly fall into local optimum. The Mean fitness function value and the best fitness function value of the algorithm in this paper are significantly improved. The standard deviation also reflects the good stability of the algorithm. The structural advantage of the bi-level structure increases the diversity of particle swarm. The lower-level working particle swarms run in parallel, which greatly reduces the too-long running time caused by the large-scale particle swarms. Meanwhile, both the upper and lower levels coordinate to achieve rational running, and the upper-level controls the lower-level effectively by the exponentially distributed learning factor. The algorithm in this paper proves that the ways to enlarge the size of the particle swarm and improve the structure of the particle swarm can effectively avoid the precocity convergence problem and local optimum problem of the PSO in later stage. To sum up, the large-scale bi-level particle swarm optimization algorithm is featured in not only high convergence accuracy, but also good optimization effects and less running time.

V. CONCLUSION

A large-scale bi-level particle swarm optimization algorithm is proposed in this paper targeting at PSO problems of bad diversity in initial stage and local optimum in later stage. It takes use of the large-scale bi-level particle swarm design to improve the particle swarm diversity in initial stage and reduce the possibility of being trapped in local optimum in later stage, thereby improving the algorithm stability. Meanwhile, it also uses the exponentially distributed learning factor to control the particle swarm coupling, which improves the computational efficiency of the algorithm. The parallel operation of the lower-level particle swarms improves the operating efficiency of the algorithm and reduces the too-long running time problem caused by the complex structure. The simulation experiments prove that the large-scale bi-level particle swarm optimization algorithm is featured in not only satisfactory optimization effects, but also improved stability.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Piscataway, NJ, USA, Nov./Dec. 1995, pp. 1942–1948.

[2] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, New York, NY, USA, Oct. 1995, pp. 39–43.

[3] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in solar photovoltaic system," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2997–3006, May 2013, doi: 10.1016/j.asoc.2012.11.033.

[4] N. Geng, D. W. Gong, and Y. Zhang, "PSO-based robot path planning for multisurvivor rescue in limited survival time," *Math. Problems Eng.*, vol. 2014, pp. 1–10, Sep. 2014, doi: 10.1155/2014/187370.

[5] X. Wang, G. Zhang, J. Zhao, H. Rong, F. Ipate, and R. Lefticaru, "A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning," *Int. J. Comput. Commun. Control*, vol. 10, no. 5, p. 732, Jun. 2015, doi: 10.15837/ijccc.2015.5.2030.

[6] M. Duan, "Short-time prediction of traffic flow based on PSO optimized SVM," in *Proc. Int. Conf. Intell. Transp., Big Data Smart City (ICITBS)*. Xiamen, China: IEEE Computer Society, Jan. 2018, pp. 41–45, doi: 10.1109/ICITBS.2018.00018.

[7] G. Li and W. Chou, "Path planning for mobile robot using self-adaptive learning particle swarm optimization," *Sci. China Inf. Sci.*, vol. 61, no. 5, pp. 263–280, May 2018, doi: 10.1007/s11432-016-9115-2.

[8] N. Himanshu and A. Burman, "Determination of critical failure surface of slopes using particle swarm optimization technique considering seepage and seismic loading," *Geotechnical Geolog. Eng.*, vol. 37, no. 3, pp. 1261–1281, Jun. 2019, doi: 10.1007/s10706-018-0683-8.

[9] U. K. Acharya and S. Kumar, "Particle swarm optimization exponential constriction factor (PSO-ECF) based channel equalization," in *Proc. 6th Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, New Delhi, India, Mar. 2019, pp. 94–97.

[10] R. Yan, T. Wang, X. Jiang, Q. Zhong, X. Huang, L. Wang, and X. Yue, "Design of high-performance plasmonic nanosensors by particle swarm optimization algorithm combined with machine learning," *Nanotechnology*, vol. 31, no. 37, Sep. 2020, Art. no. 375202, doi: 10.1088/1361-6528/ab95b8.

[11] M. G. Carneiro, R. Cheng, L. Zhao, and Y. C. Jin, "Particle swarm optimization for network-based data classification," *Neural Netw.*, vol. 110, pp. 243–255, Feb. 2019.

[12] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput. IEEE World Congr. Comput. Intell.*, Jun. 1998, pp. 69–73, doi: 10.1109/ICEC.1998.699146.

[13] K. Kentzoglanakis and M. Poole, "Particle swarm optimization with an oscillating inertia weight," in *Proc. 11th Annu. Conf. Genet. Evol. Comput. (GECCO)*, Montreal, QC, Canada, 2009, pp. 1749–1750, doi: 10.1145/1569901.1570140.

[14] W. Dong, L. Kang, and W. Zhang, "Opposition-based particle swarm optimization with adaptive mutation strategy," *Soft Comput.*, vol. 21, no. 17, pp. 5081–5090, Sep. 2017, doi: 10.1007/s00500-016-2102-5.

[15] H. Liu, X.-W. Zhang, and L.-P. Tu, "A modified particle swarm optimization using adaptive strategy," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113353, doi: 10.1016/j.eswa.2020.113353.

[16] J. G. Jiang, M. Tian, X. Q. Wang, X. P. Long, and J. Li, "Adaptive particle swarm optimization via disturbing acceleration coefficents," *J. Xidian Univ.*, vol. 39, no. 4, pp. 74–80, Aug. 2012, doi: 10.3969/j.issn.1001-2400.2012.04.014.

[17] C.-M. Yan, G.-Y. Lu, Y.-T. Liu, and X.-Y. Deng, "A modified PSO algorithm with exponential decay weight," in *Proc. 13th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, Guilin, China, Jul. 2017, pp. 239–242, do: 10.1109/FSKD.2017.8393146.

[18] C. Z. Tao and J. M. Yang, "PSO algorithm based on network neighborhood topology," *Comput. Eng.*, vol. 36, no. 19, pp. 18–20, Oct. 2010, doi: 10.3724/SP.J.1238.2010.00585.

[19] X. Zhang, H. Nguyen, X.-N. Bui, Q.-H. Tran, D.-A. Nguyen, D. T. Bui, and H. Moayedi, "Novel soft computing model for predicting blast-induced ground vibration in open-pit mines based on particle swarm optimization and XGBoost," *Natural Resour. Res.*, vol. 29, no. 2, pp. 711–721, Apr. 2020, doi: 10.1007/s11053-019-09492-7.

[20] J. Robinson, S. Sinton, and Y. Rahmat-Samii, "Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horn antenna," in *Proc. IEEE Antennas Propag. Soc. Int. Symp.*, San Antonio, TX, USA, Feb. 2002, pp. 314–317, doi: 10.1109/APS.2002.1016311.

[21] S. C. Duong, H. Kinjo, E. Uezato, and T. Yamamoto, "Particle swarm optimization with genetic recombination: A hybrid evolutionary algorithm," *Artif. Life Robot.*, vol. 15, no. 4, pp. 444–449, Dec. 2010, doi: 10.1007/s10015-010-0846-z.

[22] A. P. Engelbrecht, "Particle swarm optimization with crossover: A review and empirical analysis," *Artif. Intell. Rev.*, vol. 45, no. 2, pp. 131–165, Feb. 2016, doi: 10.1007/s10462-015-9445-7.

[23] X.-H. Chen, S.-W. Liu, J. Guan, and Q. Liu, "Study on QoS multicast routing based on ACO-PSO algorithm," in *Proc. Int. Conf. Intell. Comput. Technol. Autom.* Washington, DC, USA: IEEE Computer Society, vol. 3, May 2010, pp. 534–537, doi: 10.1109/ICICTA.2010.419.

[24] M. K. Patel, M. R. Kabat, and C. R. Tripathy, "A hybrid ACO/PSO based algorithm for QoS multicast routing problem," *Ain Shams Eng. J.*, vol. 5, no. 1, pp. 113–120, Mar. 2014, doi: 10.1016/j.asej.2013.07.005.

[25] D. Pal, P. Verma, D. Gautam, and P. Indait, "Improved optimization technique using hybrid ACO-PSO," in *Proc. 2nd Int. Conf. Next Gener. Comput. Technol. (NGCT)*, Dehradun, India, Oct. 2016, pp. 277–282, doi: 10.1109/NGCT.2016.7877428.

[26] D. X. Luo, Y. Q. Zhou, H. J. Huang, and X. Q. Wei, "Multi colony particle swarm optimization algorithm," *Comput. Eng. Appl.*, vol. 46, no. 19, pp. 51–54, 2010, doi: 10.3778/j.issn.1002-8331.2010.19.014.

[27] M. Lovbjerg, T. K. Rasmussen, and T. Krink, "Hybrid particle swarm optimiser with breeding and subpopulations," in *Proc. 3rd Genet. Evol. Comput. Conf.*, 2001, pp. 469–476.

[28] Y. M. Liu, C. L. Sui, and Q. Z. Zhang, "Dynamic multi-swarm particle swarm optimizer based on K-means clustering and its application," *Control Decis.*, vol. 26, no. 7, pp. 1019–1025, Jul. 2011, doi: 10.13195/j.cd.2011.07.61.liuym.011.

[29] Y. Chen, L. Li, H. Peng, J. Xiao, Y. Yang, and Y. Shi, "Particle swarm optimizer with two differential mutation," *Appl. Soft Comput.*, vol. 61, pp. 314–330, Dec. 2017, doi: 10.1016/j.asoc.2017.07.020.

[30] Y. Chen, L. Li, J. Xiao, Y. Yang, J. Liang, and T. Li, "Particle swarm optimizer with crossover operation," *Eng. Appl. Artif. Intell.*, vol. 70, pp. 159–169, Apr. 2018, doi: 10.1016/j.engappai.2018.01.009.

[31] J. H. Han, Z. R. Li, and Z. C. Wei, "Adaptive particle swarm optimization algorithm and simulation," *J. Syst. Simul.*, vol. 10, pp. 2969–2971, Oct. 2006, doi: 10.16182/j.cnki.joss.2006.10.070.

[32] G.-G. Wang, S. Deb, and L. D. S. Coelho, "Elephant herding optimization," in *Proc. 3rd Int. Symp. Comput. Bus. Intell. (ISCBI)*, Dec. 2015, pp. 1–5, doi: 10.1109/ISCBI.2015.8.

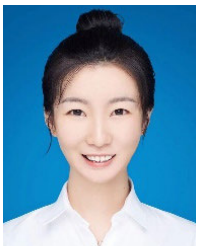
[33] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015, doi: 10.1016/j.knosys.2015.07.006.

[34] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 1995–2014, May 2015, doi: 10.1007/S00521-015-1923-Y.

[35] G. G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimization algorithm: A bio-inspired metaheuristic algorithm for global optimization problems," *Int. J. Bio-Inspired Comput*, vol. 12, no. 1, pp. 1–22, Jun. 2018, doi: 10.1504/IJBIC.2018.093328.

JIA-JIA JIANG was born in Shandong, China, in 1997. She received the B.S. degree from the Shandong University of Science and Technology, Taishan Institute of Science and Technology, China, in 2019, where she is currently pursuing the M.S. degree. Her research interests include artificial intelligence and evolutionary computation.

WEN-XUE WEI is currently an Associate Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology, China. He has published more than 30 articles and a monograph in important academic journals. His research interests include computer networking, information security, the Internet of Things engineering, and digital mining.



WAN-LU SHAO was born in Shandong, China, in 1995. She received the B.S. degree from the Shandong University of Science and Technology, China, in 2018, where she is currently pursuing the M.S. degree. Her research interests include image processing and deep learning.

YU-FENG LIANG was born in Shandong, China, in 1993. He received the B.S. degree from the Qingdao University of Science and Technology, China, in 2017. He is currently pursuing the M.S. degree with the Shandong University of Science and Technology. His research interests include artificial intelligence and deep learning.

YUAN-YUAN QU was born in Liaoning, China, in 1997. She received the B.S. degree from the Jilin Business and Technology College, China, in 2019. She is currently pursuing the M.S. degree with the Shandong University of Science and Technology. Her research interests include image processing and deep learning.