Applied Intelligence (2019) 49:335–351
https://doi.org/10.1007/s10489-018-1258-3

Enhanced particle swarm optimization with multi-swarm and multi-velocity for optimizing high-dimensional problems

Yong Ning[1] · Zishun Peng[1] · Yuxing Dai[1,2] · Daqiang Bi[3] · Jun Wang[1]

Published online: 29 August 2018

Abstract
Traditional particle swarm optimization (PSO) algorithm mainly relies on the history optimal information to guide its optimization. However, when the traditional PSO algorithm searches high-dimensional complex problems, wrong position information of the best particles can easily cause the most of the particles move toward wrong space, so the traditional PSO algorithm is easily trapped into local optimum. To improve the optimization performance of the traditional PSO algorithm, an enhanced particle swarm optimization with multi-swarm and multi-velocity (MMPSO) is proposed. It comprises three particle swarms and three velocity update methods. The information sharing of the multi-swarm with various velocity update methods in the MMPSO can quickly discover more useful global information and local information, helping prevent particles from falling into local optimum and improving optimization precision of the algorithm. The MMPSO is tested on fourteen benchmark functions, and is compared with the other improved PSO algorithms. Comparison results validate the validity and feasibility of the MMPSO to optimize high-dimensional problems.

Keywords Particle swarm optimization · High-dimensional complex problems · Information sharing · Multi-swarm · Multi-velocity

1 Introduction

Particle swarm optimization (PSO) algorithm is one of the most classical swarm intelligence algorithms, which was introduced by Kennedy and Eberhart in 1995 [9]. In order to balance the local search ability and global search ability, Shi and Eberhart introduced the inertia weight into the original PSO algorithm in 1998 [25]. Owing to its simple concept and high efficiency, traditional PSO algorithm has been widely adopted in many fields, such as micro-grids, renewable energy resources, data mining, image recognition, biology, and geology [1, 2, 10, 20–23, 26].

Traditional PSO algorithm is one-group and one-velocity algorithm, its particles search better areas mainly rely on the history optimal information of all particles, which means that the optimization process of all particles are easily affected by the wrong optimal information. When it searches high-dimensional complex problem space, there are some drawbacks with it, such as low accuracy of the optimization and low reliability of the optimization [4, 5, 7, 17, 31].

To improve the performance of the traditional PSO algorithm, the improved inertia weight methods and the improved acceleration coefficients methods are considered by many researchers. In [30], an improved PSO algorithm with high-order nonlinear time-varying inertia weight is proposed. In [29], an adaptive parameter tuning of the particle swarm optimization based on the velocity information is proposed. In this algorithm, the values of the inertia weight are changed by means of comparing the values of the average velocity and ideal velocity. In [13], an improved PSO algorithm based on the adaptive inertia weight is proposed. Its inertia weight is changed by means of the success rate of the particles towards the optimum position. In

✉ Zishun Peng
pzshnu@hnu.edu.cn

Jun Wang
junwang@hnu.edu.cn

1 College of Electrical and Information Engineering, Hunan University, Changsha 410082, China

2 College of Physics and Electronic Information Engineering, Wenzhou University, Changsha 325035, China

3 State Key Laboratory of Power System, Department of Electrical Engineering, Tsinghua University, Beijing 100084, China

[28], an improved PSO algorithm based on the dynamic adaptive inertia weight adjusting strategy and time-variant acceleration coefficients is proposed. Its inertia weight is adjusted by means of the swarm success rates, and its acceleration coefficients are nonlinear time-variant parameters. Although, these improved methods can enhance the performance of the traditional PSO algorithm, most particles of these methods are lack of a method which is jumping out of local optimum when they have been falling into local optimum.

Because the optimization ability of the improved PSO algorithm with only improved inertia weight or acceleration coefficients is limited, many researchers change the structure of the velocity update method or position update method to enhance the optimization ability of the algorithm. In [3], a chaotic concept, time varying inertia weight, and time varying acceleration coefficient are introduced into the traditional PSO algorithm to accelerate optimization and prevent the optimization falling into local optimum. However, the chaos concept is only considered in the initialization stage and local search, the global search is neglected. In fact, the global search ability of the algorithm can also help enhance optimization speed and avoid trapping into local optimum. In [27], a self-regulating PSO algorithm is proposed to achieve fast convergence and improve convergence precision. To accelerate the optimization process of the optimal particle, the optimal particle achieves self-regulation, so the inertia weight and velocity update method of the optimal particle are different with the other particles. However, with the increasing of the dimension of the problems, all particles are more and more easily falling into local optimum, accelerating the optimization process of the optimal particle cannot effectively improve this situation. In [14], an enhanced PSO algorithm incorporating a weighted particle is proposed to reduce the chance of falling into local optimum. It defines a weighted particle relying on the fitness values to provide a promising optimization direction. However, the smallest fitness values of current iteration cannot determine whether it is the global optimal values, so the weighted particle decided by the fitness values still has a risk of trapping into local optimum.

These improved PSO algorithms are lack of information sharing and acting synchronously ability, so their optimization ability will become worse when optimizing high-dimensional problems. In order to solve these problems, the coordinate optimization of the multiple swarms is considered by some researchers. In [6], an improved comprehensive learning PSO algorithm is proposed to solve the global optimization problems. It possesses two particle swarms called the master swarm and slave swarm, respectively. And these two particle swarms can share information from each other. However, this algorithm is lack of strong local optimization ability, so its optimization accuracy is worse when optimizing high-dimensional problems. In [11], a mutation mechanism and a dynamic algorithm parameters scheme are introduced into the traditional PSO algorithm to improve the global optimization ability and local optimization ability. This algorithm also divides the particle swarm into the two particle swarms, which are good sub-swarm and bad sub-swarm, respectively. Each particle swarm has the same velocity update method, and the particles of the bad sub-swarm will be gradually replaced by the better trial particles. However, only changing the structure of the inertia weight and acceleration coefficients is not enough to decrease the chance of falling into local optimum. If the particles of the good sub-swarm are in the local optimum, the particles of the bad sub-swarm can only obtain information from the local optimum, so all particles will easily trap into local optimum. In [16], a modified coevolutionary multi-swarm PSO algorithm is proposed to enhance the optimization ability of the traditional PSO algorithm. This method comprises a new velocity update method and an effective boundary constraint technique, and it adopts an information sharing strategy to achieve cooperative optimization of the multi-swarm. However, this method is lack of the test of the high-dimensional problems, and the improvement of the global optimization ability is limited by means of adopting the single velocity update method and inertial weight of linear change. In [19], a cooperative multi-swarm PSO algorithm is proposed. Its information sharing and random regrouping operation strategy of the different sub-swarms can improve the global optimization ability of the algorithm. However, this method neglects the improvement of the local optimization ability of the algorithm, so the precision of the optimization of this method still needs to be improved. In [15], an improved PSO algorithm incorporating the co-evolution strategy and multi-level region of the interest strategy is proposed (It is a multi-swarm PSO algorithm). Although the coordinate optimization of the multi-swarm with multi-velocity in this improved PSO algorithm can improve the performance of the traditional PSO algorithm, this method mainly focuses on the discrete problems, which is not suitable for the high-dimensional continuous problems. In [12], the fractional global best formation with the multi-swarms and multi-dimensional PSO are proposed to enhance the optimization performance of the PSO algorithm. However, their optimization methods are restricted by the optimal information of all particles. And they are lack of the mutation mechanism and strong local optimization ability. Therefore, they need much more particle swarms to enhance their own optimization

performance when they optimize the high-dimensional problems.

In this paper, an enhanced particle swarm optimization algorithm with multi-swarm and multi-velocity (MMPSO) is proposed to improve the optimization performance of the traditional PSO algorithm. It comprises three particle swarms and three velocity update methods. Compared to the other PSO algorithms, the proposed PSO algorithm can discover more useful information by means of the information sharing of the multiple particle swarms and improve the diversity of the optimization tendency by means of the multiple velocity update methods. Therefore, the coordinate optimization of the different particle swarms with different velocity update methods in MMPSO can strengthen its optimization precision and reliability.

The rest of the paper is organized as follows. In Section 2, the principles of the traditional PSO algorithm and MMPSO are introduced. And the optimization process of the MMPSO are elaborated. In Section 3, the selection of the optimization parameters of the MMPSO and the comparisons of the MMPSO with selected state of the art PSO algorithms over fourteen benchmark functions are studied. Finally, conclusions are given in Section 4.

2 Principles of traditional PSO algorithm and MMPSO

2.1 Traditional PSO algorithm

In the traditional PSO algorithm, each particle is regarded as an individual without quality and volume. It moves in a specific space at a certain speed through the local best of personal particle and global best of all particles, each particle continuously adjusts its speed and gradually moves toward the optimal area. In the procedure of optimization, the new velocity and position information of the particles are updated according to the following two equations [25]

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 (x_{ij}^P(t) - x_{ij}(t)) + c_2 r_2 (x_{gj}^G(t) - x_{ij}(t)) \quad (1)$$

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

where t denotes the number of current iteration, w is the inertia weight, c_1 and c_2 are the acceleration coefficients, r_1 and r_2 are the random numbers in [0, 1]. $x_{ij}(t)$ and $v_{ij}(t)$ represent the jth ($j = 1, 2, \cdots, D$) dimension of the ith ($i = 1, 2, \cdots, N$) particle's position vector and velocity vector, respectively. $x_{ij}^P(t)$ and $x_{gj}^G(t)$ represent the local best of personal particle and global best of all particles, respectively.

2.2 MMPSO

In the traditional PSO algorithm, $c_1 r_1 (x_{ij}^P(t) - x_{ij}(t))$ and $c_2 r_2 (x_{gj}^G(t) - x_{ij}(t))$ are the two primary terms to describe the movement direction of all particles. When the particles' position $x_{ij}(t)$ is coming close to the optimal position $x_{ij}^P(t)$ and $x_{gj}^G(t)$, the movement distance of most particles will decrease, so the chance of falling into local optimum will increase.

To solve this problem, the MMPSO is proposed. Its composition is shown in Fig. 1. It comprises three particle swarms and three velocity update methods. Three particle swarms consist of the main particle swarm, global auxiliary particle swarm, and local auxiliary particle swarm. Three velocity update methods consist of the velocity update method with improved inertia weight, improved velocity update method with constriction factor, and random velocity update method. The particles with large fitness value of the

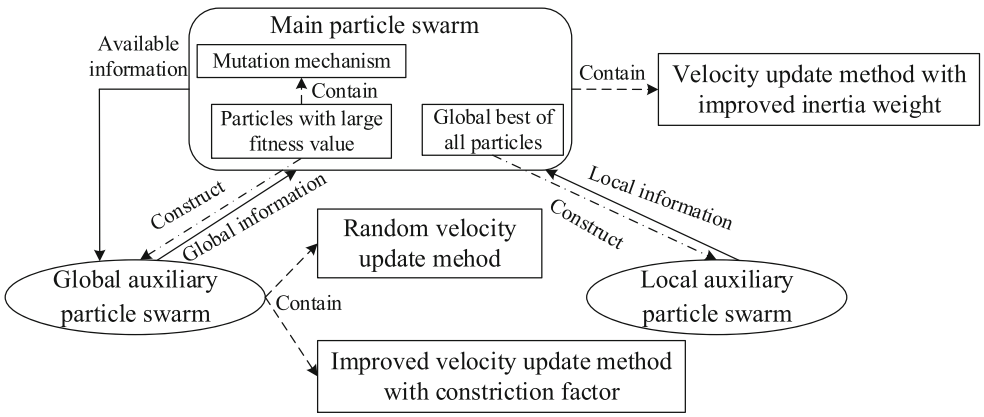


Fig. 1 Composition of MMPSO

main particle swarm construct the global auxiliary particle swarm and have the mutation mechanism. And the global best of all particles of the main particle swarm construct the local auxiliary particle swarm. The velocity update method with improved inertia weight is adopted by the main particle swarm. The improved velocity update method with constriction factor and random velocity update method are adopted by the global auxiliary particle swarm, which plays a role of searching large range area. The local auxiliary particle swarm has no velocity update method, which plays a role of searching small range area based on the current global optimal position. The main particle swarm also provides available information to the global auxiliary particle swarm during the whole optimization iteration. After a predefined iteration, the global auxiliary particle swarm provides the better global information to the main particle swarm. Then, the local auxiliary particle swarm provides the better local information to the main particle swarm after the global auxiliary particle swarm stops working, and this process continues until the maximum iteration is finished. Its detail composition is introduced as follows.

2.2.1 Multi-swarm

Various particle swarms can help to improve the diversity of the particles. Their detail information will be described in the following.

a) Main particle swarm

The main particle swarm is an original particle swarm, which constructs the global auxiliary particle swarm and local auxiliary particle swarm. And it also obtains better information from these two auxiliary particle swarms. Because the particles with large fitness value in the main particle swarm of the current iteration do not affect the optimal particles, variation of these particles can be expected to increase the diversity of the particles. Its formula is written as

$$x_{mnew}(t) = (1.5 - 3r)x_w(t) \tag{3}$$

where x_w is the position vector of the particles with large fitness value, and x_{mnew} is the position vector of the mutant particles. r is the random numbers in [0, 1].

b) Global auxiliary particle swarm

The global auxiliary particle swarm is adopted to search the area in the global level. Its composition is shown in Fig. 2. The square represents the unsorted particles of the main particle swarm. The white circle represents the sorted particles of the main particle swarm. The black circle represents the particles of the global auxiliary particle swarm. N and s are the number of particles, and there is $N = s + s$. First, the particles of the main particle swarm are sorted from the largest fitness value to the smallest fitness value. Then, the particles from the largest fitness value to the middle fitness value form a new swarm. Finally, these particles are duplicated to form the global auxiliary particle swarm. Therefore, it has the same numbers of the particles with the main particle swarm.

The position update method of the main particle swarm and global auxiliary particle swarm are the same as those of the (2).

c) Local auxiliary particle swarm

In the end of the optimization iteration, the global best of all particles is almost in the global best area, so the second auxiliary swarm called the local auxiliary particle swarm is adopted to search the better

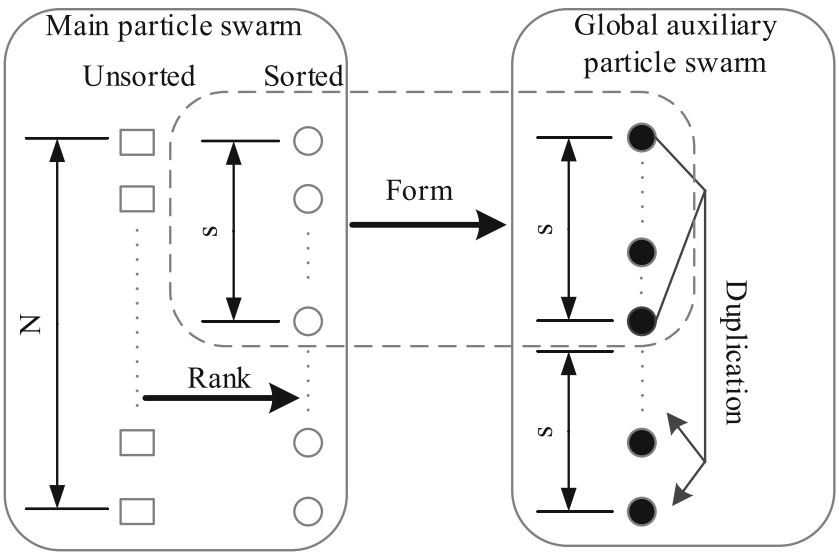


Fig. 2 Formation of global auxiliary particle swarm

information nearby the global best of all particles. Its formula is written as

$$x_{2ij}(t) = x_{gj}^{G}(t)+(0.5-r)x_{gj}^{G}(t) \tag{4}$$

2.2.2 Multi-velocity

Various velocity update methods can help to improve the diversity of the optimization tendencies. Their detail information will be described in the following.

a) Velocity update method with improved inertia weight

The velocity update method of the main particle swarm is the same as that of the (1). However, the inertia weight w in this paper is a nonlinear time-varying parameter, which is suitable for the nonlinear optimization of the main particle swarm. The w is defined as follows

$$w = w_s - (w_s - w_e)\left(\frac{w_s}{w_e}\right)^{k(1-\frac{t}{\max t})} \tag{5}$$

where k, w_s, w_e, and $\max t$ are the constant coefficient, maximal inertia weight, minimal inertia weight, and maximal iteration, respectively.

b) Improved velocity update method with constriction factor and random velocity update method

Two velocity update methods are adopted by the global auxiliary particle swarm. One is the improved velocity update method with constriction factor and another is the random velocity update method. Only one of these two velocity update methods are selected with a certain probability in each iteration.

Compared to the traditional PSO algorithm, the PSO algorithm with a constriction factor has stronger global exploration ability [32]. Therefore, this velocity update method can be introduced into the global auxiliary particle swarm to enhance its global optimization performance. Its formula is written as

$$\begin{aligned} v_{1ij}(t+1) = {} & \psi(v_{1ij}(t) + c_3 r_3(x_{1ij}^{P}(t) \\ & -x_{1ij}(t))+c_4 r_4(x_{1dj}{}^{G}(t) - x_{1ij}(t))) \end{aligned} \tag{6}$$

where ψ is the constriction factor, c_3 and c_4 are the acceleration coefficients, r_3 and r_4 are the random numbers in [0, 1]. $x_{1ij}(t)$ and $v_{1ij}(t)$ are the particle's position vector and velocity vector, respectively. $x_{1ij}^{P}(t)$ and $x_{1dj}^{G}(t)$ are the local best of personal particle and global best of all particles, respectively.

The constriction factor ψ is defined as

$$\psi = \begin{cases} \dfrac{2}{\left|2-\phi-\sqrt{\phi^2-4\phi}\right|}, & \phi = c_3 r_3 + c_4 r_4 \geq 4 \\ 1, & \phi = c_3 r_3 + c_4 r_4 < 4 \end{cases} \tag{7}$$

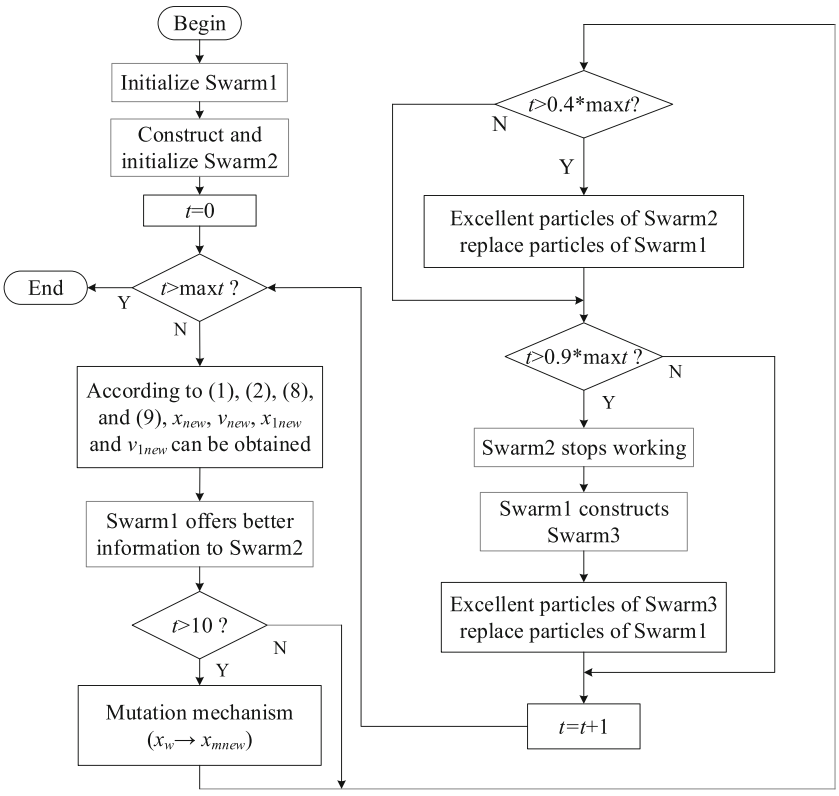


Fig. 3 Optimization flowchart of MMPSO

Based on the velocity update method with constriction factor, a large disturbance is added into this velocity update method to further enhance the global optimization ability of the global auxiliary particle swarm. Its formula is written as

$$\begin{aligned} v_{1ij}(t+1) = \psi(v_{1ij}(t) + c_3 r_3(x^P_{1ij}(t) \\ -x_{1ij}(t)) + c_4 r_4(x^G_{1dj}(t) - x_{1ij}(t))) \\ + \frac{(0.5 - r_5)(x^G_{1dj}(t) - x^P_{1ij}(t))}{1 + e^{1-F_i}} \end{aligned} \tag{8}$$

where F_i is the fitness value of current iteration, r_5 is random number in [0, 1]. $(0.5 - r_5)(x^G_{1dj}(t) - x^P_{1ij}(t))/(1 + e^{(1-F_i)})$ represents the disturbance, which has some probability to change the direction of optimization. In the early iteration, the local best of personal particle and global best of all particles are far from each other and the fitness values of the particles are also large, so the values of the velocity equation are large, which means that the global search capability of the global auxiliary particle swarm is strong. The values of the velocity equation are decreasing with the increasing of the iteration, which means that the global search ability of the global auxiliary particle swarm is gradually becoming weak.

The random velocity update method is introduced to enlarge the global search area of the global auxiliary particle swarm without affected by the local best of personal particle and global best of all particles. It is defined as

$$v_{1ij}(t+1) = v_{1ij}(t) + F(t) \tag{9}$$

where

$$\begin{aligned} F(t) = [(0.5-r)(x_{1i1}(t) - V_{\max})(0.5-r)(x_{1i2}(t) - V_{\max}) \\ \cdots (0.5 - r)(x_{1iD}(t) - V_{\max})] \end{aligned} \tag{10}$$

The random velocity update method has a character of extraordinary random. Moderate random mechanism can effectively diffuse particles and prevent optimization from falling into local optimum. However, too strong random mechanism may make the optimization of the global auxiliary particle swarm becomes blind and prevent the global auxiliary particle swarm from finding the global optimal area. Therefore, the probability of the random velocity update method should be set to a medium value.

In the end of the optimization iteration, the global optimization ability of the global auxiliary particle swarm is becoming weak. Therefore, the global auxiliary particle swarm can stop working to reduce the computational amount of the MMPSO.

The pseudo-code of the main part of the MMPSO is written in the Algorithm 1.

Algorithm 1 Pseudo-code of main part of MMPSO

```
Initialize the particles and velocity of the main particle swarm and global auxiliary particle swarm
while t < max t do
    The value of the w is calculated by the (5)
    for j = 1 : D do
        The value of the velocity of the main particle swarm is updated by the (1)
        The value of the position of the main particle swarm is updated by the (2)
        if r > 0.01 then
            The value of the velocity of the global auxiliary particle swarm is updated by the (8)
        else
            The value of the velocity of the global auxiliary particle swarm is updated by the (9)
        end if
        The value of the position of the global auxiliary particle swarm is updated by the (2)
        if t > 10 then
            The mutation of the particles with the large fitness values in the main particle swarm occur (The formula of the mutation is the (3))
        end if
        if max t * 0.9 ≥ t and t > max t * 0.4 then
            Excellent particles of the global auxiliary particle swarm replace the particles of the main particle swarm
        end if
        if t > max t * 0.9 then
            The local auxiliary particle swarm is constructed by the (4)
            Excellent particles of the global auxiliary particle swarm replace the particles of the main particle swarm
        end if
    end for
end while
```

2.3 Optimization flowchart of MMPSO

After several experiments, the optimal threshold of the local auxiliary particle swarm is set to 0.9*maxt and the threshold of the better information replacement of the global auxiliary particle swarm is set to 0.4*maxt can make the MMPSO achieve better optimization performance. The optimization flowchart of the MMPSO is illustrated in Fig. 3. Swarm1 represents the main particle swarm, Swarm2 represents the global auxiliary particle swarm, and Swarm3 represents the local auxiliary particle swarm. x_{new} and v_{new} are the updated position vector and velocity vector of the main particle swarm, respectively. x_{1new} and v_{1new} are the

updated position vector and velocity vector of the global auxiliary particle swarm, respectively. The key steps of optimization process of the MMPSO are listed as follows.

a) Randomly generate D-dimensions N-particles. Each particle is associated with a position vector $x_{ij}(0)$ and a velocity vector $v_{ij}(0)$. $x_{gj}^{G}(0)$ and $x_{gj}^{G}(0)$ are updated by the initial fitness value. The particles with large fitness value of the main particle swarm construct the global auxiliary particle swarm, and each particle is associated with a position vector $x_{1ij}(0)$ and a velocity vector $v_{1ij}(0)$. $x_{1ij}^{P}(0)$ and $x_{1dj}^{G}(0)$ are obtained from $x_{ij}^{P}(0)$ and $x_{gj}^{G}(0)$. In the process of the optimization, the main particle swarm offers better information to the global auxiliary particle swarm in each iteration. The position vector $x_{ij}(t)$ and velocity vector $v_{ij}(t)$ of the main particle swarm are updated according to (1) and (2). The position vector $x_{1ij}(t)$ and velocity vector $v_{1ij}(t)$ of the global auxiliary particle swarm are updated according to (2) and (8) or (9).

b) When the number of optimization iteration ranges from 11 to $0.4 * \max t$, the particles with large fitness value of the main particle swarm have mutation mechanism. In this phase, the main particle swarm is not received the better information from the global auxiliary particle swarm.

Table 1 Benchmark functions

	Benchmark functions	Domain	f_{min}
Sphere Model Function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[-100,100]	0
Generalized Rosenbrock's Function	$f_2(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	[-2.048,2.048]	0
Ackley Function	$f_3(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$ -exp$(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	[-32,32]	0
Griewanks's function	$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0
Generalized Rastrigin's Function	$f_5(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	[-5.12,5.12]	0
Non-continuous Rastrigin's Function	$f_6(x) = \sum_{i=1}^{D} \left[y_i^2 - 10\cos(2\pi y_i) + 10\right],\ y_i = \begin{cases} x_i, & \lvert x_i\rvert < 0.5 \\ \frac{round(2x_i)}{2}, & \lvert x_i\rvert \geq 0.5 \end{cases}$	[-5.12,5.12]	0
Weierstrass Function	$f_7(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left(a^k\cos(2\pi b^k(x_i + 0.5))\right)\right) - D\sum_{k=0}^{k_{\max}}(a^k\cos(\pi b^k)),$ $a = 0.5,\ b = 3,\ k_{\max} = 20$	[-0.5,0.5]	0
Schwefel Function	$f_8(x) = 418.9828D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	0
Rotated Rastrigin's Function	$f_9(x) = \sum_{i=1}^{D} \left[y_i^2 - 10\cos(2\pi y_i) + 10\right],\ y_i = M * x_i$	[-5.12,5.12]	0
Rotated Ackley Function	$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi y_i) + 20 + e,$ $y_i = M * x_i$	[-32,32]	0
Rotated Griewanks's Function	$f_{11}(x) = \sum_{i=1}^{D} \frac{y_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{y_i}{\sqrt{i}}) + 1,\ y_i = M * x_i$	[-600,600]	0
Rotated Schwefel Function	$f_{12}(x) = 418.9828D - \sum_{i=1}^{D} z_i,\ z_i = \begin{cases} y_i \sin(\sqrt{\lvert y_i\rvert}), & \lvert y_i\rvert \leq 500 \\ 0, & \text{otherwise} \end{cases},$ $y_i = y_i' + 420.96,\ y_i' = M(x_i - 420.96)$	[-500,500]	0
Rotated Non-continuous Rastriginr's Function	$f_{13}(x) = \sum_{i=1}^{D} \left[z_i^2 - 10\cos(2\pi z_i) + 10\right],$ $z_i = \begin{cases} y_i, & \lvert y_i\rvert < 0.5 \\ \frac{round(2y_i)}{2}, & \lvert y_i\rvert \geq 0.5 \end{cases},\ y_i = M * x_i$	[-5.12,5.12]	0
Rotated Weierstrass Function	$f_{14}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left(a^k\cos(2\pi b^k(y_i + 0.5))\right)\right) - D\sum_{k=0}^{k_{\max}}(a^k\cos(\pi b^k)),$ $a = 0.5,\ b = 3,\ k_{\max} = 20,\ y_i = M * x_i$	[-0.5,0.5]	0

c) When the number of optimization iteration ranges from $(0.4*\max t+1)$ to $0.9*\max t$, the global auxiliary particle swarm provides the better global information to the main particle swarm, which is the particles with smaller fitness value of the global auxiliary particle swarm replace the particles with larger fitness value of the main particle swarm in the same position. In this stage, the main particle swarm also has the mutation mechanism.

d) When the number of optimization iteration is larger than $0.9 * \max t$, the global auxiliary particle swarm stops working. Meanwhile, the local auxiliary particle swarm is constructed by the global best of all particles of the main particle swarm according to (3). It provides the better local information to the main particle swarm. In this stage, the main particle swarm still has mutation mechanism.

e) If the iterative termination condition is satisfied, the optimization is ended.

Table 2 Different parameters of MMPSO

Goups	k	c_3	c_4
Set 1	10	2.05	2.05
Set 2	10	3.05	3.05
Set 3	10	3.05	1.05
Set 4	10	1.05	3.05
Set 5	1	2.05	2.05
Set 6	1	3.05	3.05
Set 7	1	3.05	1.05
Set 8	1	1.05	3.05
Set 9	1.00×10^{-2}	2.05	2.05
Set 10	1.00×10^{-2}	3.05	3.05
Set 11	1.00×10^{-2}	3.05	1.05
Set 12	1.00×10^{-2}	1.05	3.05
Set 13	1.00×10^{-4}	2.05	2.05
Set 14	1.00×10^{-4}	3.05	3.05
Set 15	1.00×10^{-4}	3.05	1.05
Set 16	1.00×10^{-4}	1.05	3.05

3 Experimental results

3.1 Benchmark functions

The experiment is performed in a computer with a dual core CPU of 2.8 GHz and a RAM of 4 GB, and the program is written in Matlab 2015b. To demonstrate the performance of the MMPSO, fourteen benchmark functions are selected for the test [6]. These functions have different characteristics, which can be used to test the algorithm's abilities to optimize problems. Their function names, analytic expressions, domain, and minimum values of the functions (f_{min}) are listed in Table 1. In Table 1, M is the orthogonal matrix.

3.2 Parameter selection of MMPSO

Because the position update method and velocity update method of the main particle swarm are the same as that of the traditional PSO algorithm. Therefore, according to [8, 14, 24], c_1, c_2, w_s, and w_e can be set as 2, 2, 1.62, and 0.52, respectively.

However, the parameter k determines the variation of the nonlinear time-varying inertia weight, and parameters c_3 and c_4 determine the step length of search of the global auxiliary particle swarm. To find optimum values, sixteen kinds of parameter combinations based on the MMPSO are compared. They are denoted as Set 1−Set 16, and listed in Table 2.

MMPSO with different parameters are set with the same population size of 30, and the termination criterion allowed in each run is a maximum number of 2×10^5 function evaluations (FEs). The dimension (D) of each benchmark function is 30, and each algorithm is tested 30 times independently on each benchmark function. T-test is employed to rigorously compare the MMPSO of one group parameters with the MMPSO of the other parameters selection [18]. Comparison results are listed in Table 3, including the mean values (Mean), standard deviation values (Std), and t-value. In the Table 3, ♯ means that the difference between two samples is significant at level $\alpha = 0.05$, the best results are written in bold font.

As shown in Table 3, different parameters of the MMPSO have the same optimization accuracy and reliability on functions f_1, $f_4 - f_7$, $f_9 - f_{11}$, f_{13}, and f_{14}. When the MMPSO with different parameters are optimizing function f_2, the differences between the Set 2 and Set 1, Set 3, Set 5, Set 7, Set 9, Set 11, Set 13, Set 15, and Set 16 are statistically significant, so the optimization precision and reliability of MMPSO with the Set 2 are better than those of these nine sets of parameters. When the MMPSO with different parameters are optimizing function f_3, the differences between the Set 1 and other fifteen sets of parameters are not statistically significant, so their optimization precision and reliability are close. When the MMPSO with different parameters are optimizing function f_8, the differences between the Set 6 and Set 14 are statistically significant, so the optimization precision and reliability of MMPSO with the Set 6 are better than those of the Set 14. When the MMPSO of different parameters are optimizing function f_{10}, the differences between the Set 2 and other fifteen sets of parameters are not statistically significant, so their optimization precision and reliability are close. When the MMPSO with different parameters are optimizing function

Table 3 Optimization performance of MMPSO with different parameters

Function		Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8
f_1	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_2	Mean	2.59×10^{-6}	$\mathbf{1.42 \times 10^{-7}}$	1.86×10^{-6}	4.98×10^{-7}	1.83×10^{-6}	3.05×10^{-7}	2.36×10^{-6}	2.77×10^{-7}
	Std	5.92×10^{-6}	$\mathbf{2.37 \times 10^{-7}}$	2.72×10^{-6}	1.32×10^{-6}	2.92×10^{-6}	7.44×10^{-7}	4.63×10^{-6}	6.04×10^{-7}
	t-value	2.263♯	0	3.447♯	1.454	3.156♯	1.143	2.620♯	1.14
f_3	Mean	1.36×10^{-15}	$\mathbf{8.88 \times 10^{-16}}$	1.24×10^{-15}	1.13×10^{-15}	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$
	Std	1.54×10^{-15}	$\mathbf{1.00 \times 10^{-31}}$	1.43×10^{-15}	9.01×10^{-16}	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$
	t-value	1.679	0	1.348	1.471	0	0	0	0
f_4	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_5	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_6	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_7	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_8	Mean	$4.56 \times 10^{+0}$	4.01×10^{-2}	$0.28 \times 10^{+0}$	$0.13 \times 10^{+0}$	$0.29 \times 10^{+0}$	$\mathbf{3.82 \times 10^{-4}}$	$0.53 \times 10^{+0}$	3.12×10^{-2}
	Std	$1.91 \times 10^{+1}$	$0.20 \times 10^{+0}$	$1.53 \times 10^{+0}$	$0.73 \times 10^{+0}$	$1.22 \times 10^{+0}$	$\mathbf{1.41 \times 10^{-11}}$	$1.76 \times 10^{+0}$	$0.10 \times 10^{+0}$
	t-value	1.308	1.088	1.001	0.973	1.3	0	1.648	1.688
f_9	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_{10}	Mean	1.44×10^{-14}	$\mathbf{8.88 \times 10^{-16}}$	2.80×10^{-12}	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$
	Std	6.40×10^{-14}	$\mathbf{1.00 \times 10^{-31}}$	8.57×10^{-12}	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$
	t-value	1.156	0	1.789	0	0	0	0	0
f_{11}	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_{12}	Mean	6.26×10^{-4}	5.93×10^{-4}	5.97×10^{-4}	6.09×10^{-4}	4.07×10^{-4}	3.88×10^{-4}	4.16×10^{-4}	5.45×10^{-4}
	Std	8.89×10^{-5}	1.18×10^{-4}	1.16×10^{-4}	1.03×10^{-4}	5.89×10^{-5}	2.10×10^{-5}	1.75×10^{-4}	7.64×10^{-4}
	t-value	15.033♯	9.794♯	10.152♯	12.071♯	2.325♯	1.565	1.064	1.169
f_{13}	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_{14}	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
		Set 9	Set 10	Set 11	Set 12	Set 13	Set 14	Set 15	Set 16
f_1	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_2	Mean	$4.28 \times 10^{+0}$	4.84×10^{-6}	$4.27 \times 10^{+0}$	$0.13 \times 10^{+0}$	$3.69 \times 10^{+0}$	3.86×10^{-6}	$2.98 \times 10^{+0}$	1.01×10^{-5}
	Std	$8.71 \times 10^{+0}$	1.51×10^{-5}	$8.68 \times 10^{+0}$	$0.73 \times 10^{+0}$	$8.06 \times 10^{+0}$	1.04×10^{-5}	$7.50 \times 10^{+0}$	1.35×10^{-5}
	t-value	2.691♯	1.704	2.694♯	0.975	2.508♯	1.958	2.176♯	4.040♯

Table 3 (continued)

Function		Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8
f_3	Mean	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$
	Std	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$
	t-value	0	0	0	0	0	0	0	0
f_4	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_5	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_6	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_7	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_8	Mean	$4.28 \times 10^{+0}$	$1.19 \times 10^{+2}$	$0.19 \times 10^{+0}$	$4.16 \times 10^{+0}$	$1.19 \times 10^{+2}$	$0.17 \times 10^{+0}$	$0.20 \times 10^{+0}$	$1.05 \times 10^{+2}$
	Std	$2.16 \times 10^{+1}$	$6.49 \times 10^{+2}$	$0.78 \times 10^{+0}$	$2.16 \times 10^{+1}$	$6.49 \times 10^{+2}$	$0.43 \times 10^{+0}$	$0.56 \times 10^{+0}$	$5.72 \times 10^{+2}$
	t-value	1.085	1.004	1.332	1.055	1.004	2.161♯	1.952	1.005
f_9	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_{10}	Mean	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$	$\mathbf{8.88 \times 10^{-16}}$
	Std	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$	$\mathbf{1.00 \times 10^{-31}}$
	t-value	0	0	0	0	0	0	0	0
f_{11}	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_{12}	Mean	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{3.82 \times 10^{-4}}$	3.50×10^{-3}	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{3.82 \times 10^{-4}}$	$\mathbf{3.82 \times 10^{-4}}$	1.00×10^{-3}
	Std	8.92×10^{-13}	9.17×10^{-13}	8.92×10^{-13}	1.71×10^{-2}	9.17×10^{-13}	$\mathbf{8.72 \times 10^{-13}}$	9.23×10^{-13}	3.60×10^{-3}
	t-value	0	0	0	0.999	0	0	0	0.94
f_{13}	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0
f_{14}	Mean	0	0	0	0	0	0	0	0
	Std	0	0	0	0	0	0	0	0
	t-value	0	0	0	0	0	0	0	0

f_{12}, the differences between the Set 14 and Set 1, Set 2, and Set 3 - Set 5 are statistically significant, so the optimization precision and reliability of MMPSO with the Set 14 are better than those of these five sets of parameters. Synthesizing the performance of each set of parameters, the parameters can be selected as follow: $k = 1.00$, $c_3 = 3.05$, and $c_4 = 3.05$.

3.3 Comparisons of different PSO algorithms

To demonstrate the excellent convergence performance of the MMPSO in various complex problem spaces, fourteen benchmark functions are selected with a dimension of 10, 30, and 100 for test. The population sizes of the MMPSO are 5, 20, and 30, respectively. The population sizes of the PCLPSO are 5, 10, and 15, respectively. And the population sizes of the PSO-CF, UPSO, FDR-PSO, and CPSO-H are 10, 40, and 60, respectively. The maximum numbers of the function evaluation are 3×10^4, 2×10^5, and 3×10^5, respectively. The performance results of the PCLPSO, CLPSO, PSO-CF, UPSO, FDR-PSO, and CPSO-H are presented in [6]. T-test is employed to rigorously compare the MMPSO with the other PSO algorithm. In Tables 4, 5, and 6, ♯ means that the

Table 4 Mean and standard deviation values of the fourteen benchmark functions for $10-D$ problems

Function		MMPSO	PCLPSO	CLPSO	PSO-CF	UPSO	FDR-PSO	CPSO-H
f_1	Mean	**0**	4.29×10^{-57}	5.15×10^{-29}	9.84×10^{-105}	9.84×10^{-118}	2.21×10^{-90}	4.98×10^{-45}
	Std	**0**	1.17×10^{-56}	2.16×10^{-28}	4.21×10^{-104}	3.56×10^{-117}	9.88×10^{-90}	1.00×10^{-44}
	t-value	0	2.008♯	1.305	1.28	1.513	1.225	2.728♯
f_2	Mean	$\mathbf{8.70 \times 10^{-7}}$	5.84×10^{-1}	$2.46 \div 10^{+0}$	6.98×10^{-1}	$1.40 \times 10^{+0}$	8.67×10^{-1}	$1.53 \times 10^{+0}$
	Std	$\mathbf{1.80 \times 10^{-6}}$	$1.59 \times 10^{+0}$	$1.70 \times 10^{+0}$	$1.46 \times 10^{+0}$	$1.88 \times 10^{+0}$	$1.63 \times 10^{+0}$	$1.70 \times 10^{+0}$
	t-value	0	2.012♯	7.926♯	2.619♯	4.079♯	2.913♯	4.930♯
f_3	Mean	$\mathbf{8.88 \times 10^{-16}}$	4.00×10^{-15}	4.32×10^{-14}	9.18×10^{-1}	$1.33 \times 10^{+0}$	3.18×10^{-14}	1.49×10^{-14}
	Std	$\mathbf{1.00 \times 10^{-31}}$	2.41×10^{-30}	2.55×10^{-14}	$1.01 \times 10^{+0}$	$1.48 \times 10^{+0}$	6.40×10^{-14}	6.97×10^{-15}
	t-value	0	$7.067 \times 10^{+15}$♯	9.088♯	4.978♯	4.922♯	2.646♯	11.011♯
f_4	Mean	**0**	6.59×10^{-4}	4.56×10^{-3}	1.19×10^{-1}	1.04×10^{-1}	9.24×10^{-2}	4.07×10^{-2}
	Std	**0**	2.50×10^{-3}	4.81×10^{-3}	7.11×10^{-2}	7.10×10^{-2}	5.61×10^{-2}	2.80×10^{-2}
	t-value	0	1.444	5.193♯	9.167♯	8.023♯	9.021♯	7.962♯
f_5	Mean	**0**	**0**	**0**	$1.25 \times 10^{+1}$	$1.17 \times 10^{+1}$	$7.51 \times 10^{+0}$	0
	Std	**0**	**0**	**0**	$5.17 \times 10^{+0}$	$6.11 \times 10^{+0}$	$3.05 \times 10^{+0}$	0
	t-value	0	0	0	13.243♯	10.488♯	13.443♯	0
f_6	Mean	**0**	**0**	**0**	$1.20 \times 10^{+1}$	$5.85 \times 10^{+0}$	$3.35 \times 10^{+0}$	2.00×10^{-1}
	Std	**0**	**0**	**0**	$4.99 \times 10^{+0}$	$3.15 \times 10^{+0}$	$2.01 \times 10^{+0}$	4.10×10^{-1}
	t-value	0	0	0	13.172♯	10.172♯	9.129♯	2.672♯
f_7	Mean	**0**	**0**	**0**	6.69×10^{-1}	$1.14 \times 10^{+0}$	3.01×10^{-3}	1.07×10^{-15}
	Std	**0**	**0**	**0**	7.17×10^{-1}	$1.17 \times 10^{+0}$	7.20×10^{-3}	1.67×10^{-15}
	t-value	0	0	0	5.111♯	5.337♯	2.290♯	3.509♯
f_8	Mean	6.10×10^{-3}	1.27×10^{-4}	**0**	$3.20 \times 10^{+2}$	$1.08 \times 10^{+3}$	$8.51 \times 10^{+2}$	$2.13 \times 10^{+2}$
	Std	1.56×10^{-2}	2.78×10^{-13}	**0**	$1.85 \times 10^{+2}$	$2.68 \times 10^{+2}$	$2.76 \times 10^{+2}$	$1.41 \times 10^{+2}$
	t-value	0	2.097♯	2.142♯	9.474♯	22.072♯	16.888♯	8.274♯
f_9	Mean	**0**	$2.61 \times 10^{+0}$	$5.97 \times 10^{+0}$	$1.44 \times 10^{+1}$	$1.52 \times 10^{+1}$	$9.25 \times 10^{+0}$	$2.67 \times 10^{+1}$
	Std	**0**	$4.22 \times 10^{+0}$	$2.88 \times 10^{+0}$	$6.04 \times 10^{+0}$	$5.25 \times 10^{+0}$	$2.50 \times 10^{+0}$	$1.06 \times 10^{+1}$
	t-value	0	3.388♯	11.354♯	13.058♯	15.858♯	20.266♯	13.796♯
f_{10}	Mean	$\mathbf{8.88 \times 10^{-16}}$	8.28×10^{-9}	3.56×10^{-5}	$1.19 \times 10^{+0}$	$1.00 \times 10^{+0}$	1.40×10^{-1}	$1.36 \times 10^{+0}$
	Std	$\mathbf{1.00 \times 10^{-31}}$	4.53×10^{-8}	1.57×10^{-4}	$1.13 \times 10^{+0}$	9.27×10^{-1}	4.38×10^{-1}	8.85×10^{-1}
	t-value	0	1.001	1.242	5.768♯	5.909♯	1.751	8.417♯
f_{11}	Mean	**0**	5.50×10^{-3}	4.50×10^{-2}	1.38×10^{-1}	7.76×10^{-2}	1.44×10^{-1}	1.20×10^{-1}
	Std	**0**	1.37×10^{-2}	3.08×10^{-2}	1.07×10^{-1}	6.40×10^{-2}	7.84×10^{-2}	8.07×10^{-2}
	t-value	0	2.199♯	8.002♯	7.064♯	6.641♯	10.060♯	8.145♯
f_{12}	Mean	$\mathbf{3.39 \times 10^{-5}}$	$6.48 \times 10^{+2}$	$1.14 \times 10^{+2}$	$1.19 \times 10^{+3}$	$1.27 \times 10^{+3}$	$1.07 \times 10^{+3}$	$9.67 \times 10^{+2}$
	Std	$\mathbf{5.18 \times 10^{-5}}$	$2.58 \times 10^{+2}$	$1.28 \times 10^{+2}$	$4.23 \times 10^{+2}$	$2.29 \times 10^{+2}$	$2.23 \times 10^{+2}$	$3.67 \times 10^{+2}$
	t-value	0	13.757♯	4.878♯	15.409♯	30.376♯	26.281♯	14.432♯
f_{13}	Mean	**0**	$3.26 \times 10^{+0}$	$5.44 \times 10^{+0}$	$1.53 \times 10^{+1}$	$1.47 \times 10^{+1}$	$1.07 \times 10^{+1}$	$1.90 \times 10^{+1}$
	Std	**0**	$3.32 \times 10^{+0}$	$1.39 \times 10^{+0}$	$6.38 \times 10^{+0}$	$6.53 \times 10^{+0}$	$3.86 \times 10^{+0}$	$9.05 \times 10^{+0}$
	t-value	0	5.378♯	21.436♯	13.135♯	12.330♯	15.183♯	11.499♯
f_{14}	Mean	**0**	2.75×10^{-2}	3.72×10^{-10}	$2.17 \times 10^{+0}$	$2.61 \times 10^{+0}$	3.34×10^{-1}	$4.35 \times 10^{+0}$
	Std	**0**	4.66×10^{-2}	4.40×10^{-10}	$1.30 \times 10^{+0}$	9.48×10^{-1}	3.90×10^{-1}	$1.35 \times 10^{+0}$
	t-value	0	3.232♯	4.631♯	9.143♯	15.080♯	4.691♯	17.649♯

difference between two samples is significant at level $\alpha = 0.05$, the best results are written in bold font.

Table 4 shows the mean and standard deviation values of the fourteen benchmark functions for 10-D problems. MMPSO, PCLPSO, and CLPSO have the same mean and standard deviation values on functions f_5, f_6, and f_7, the differences between the MMPSO and PCLPSO and CLPSO are not statistically significant, so their optimization accuracy and reliability are close when optimizing these functions. However, the mean and standard deviation values

Table 5 Mean and standard deviation values of the fourteen benchmark functions for $30-D$ problems

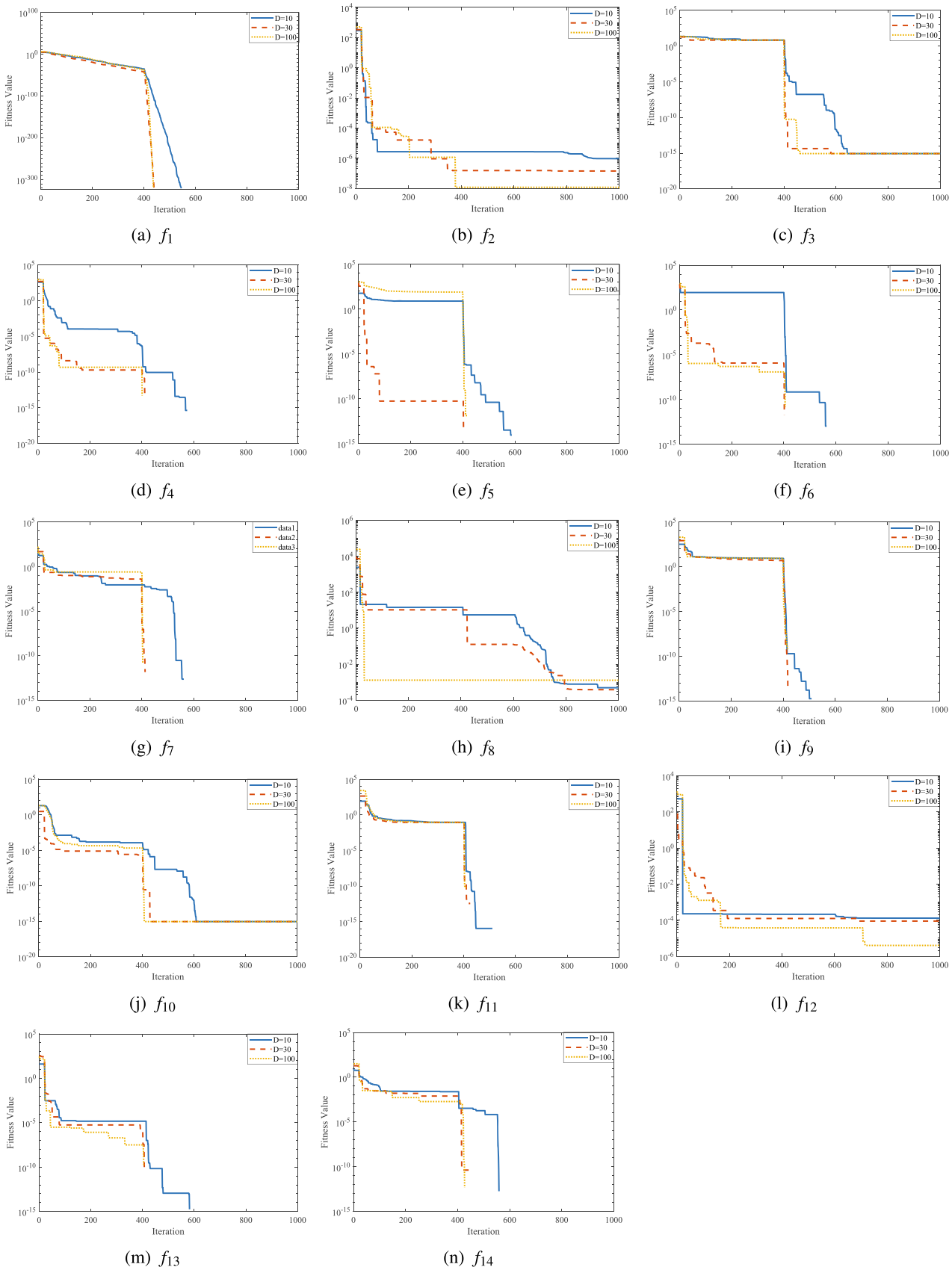
Function		MMPSO	PCLPSO	CLPSO	PSO-CF	UPSO	FDR-PSO	CPSO-H
f_1	Mean	**0**	2.87×10^{-28}	4.46×10^{-14}	5.88×10^{-100}	4.17×10^{-87}	4.88×10^{-102}	1.16×10^{-113}
	Std	**0**	6.84×10^{-28}	1.73×10^{-14}	5.40×10^{-100}	3.15×10^{-87}	1.53×10^{-101}	2.92×10^{-113}
	t-value	0	18.496♯	2.298♯	5.964♯	7.251♯	1.747	2.176♯
f_2	Mean	$\mathbf{1.01 \times 10^{-6}}$	$6.63 \times 10^{+0}$	$2.10 \times 10^{+1}$	$1.11 \times 10^{+1}$	$1.51 \times 10^{+1}$	$5.39 \times 10^{+0}$	$7.08 \times 10^{+0}$
	Std	$\mathbf{2.65 \times 10^{-6}}$	$1.17 \times 10^{+1}$	$2.98 \times 10^{+0}$	$1.81 \times 10^{+0}$	8.14×10^{-1}	$1.76 \times 10^{+0}$	$8.01 \times 10^{+0}$
	t-value	0	3.104♯	3.860♯	33.590♯	101.605♯	16.774♯	4.841♯
f_3	Mean	$\mathbf{8.88 \times 10^{-16}}$	2.05×10^{-14}	0	$1.12 \times 10^{+0}$	1.22×10^{-15}	2.84×10^{-14}	4.93×10^{-14}
	Std	$\mathbf{1.00 \times 10^{-31}}$	6.29×10^{-15}	0	8.65×10^{-1}	3.16×10^{-15}	4.10×10^{-15}	1.10×10^{-14}
	t-value	0	17.078♯	$4.86410 + 16$♯	7.092♯	0.576	36.754♯	24.106♯
f_4	Mean	**0**	9.22×10^{-12}	3.14×10^{-10}	2.06×10^{-2}	1.66×10^{-3}	1.01×10^{-2}	3.63×10^{-2}
	Std	**0**	4.25×10^{-11}	4.64×10^{-10}	1.90×10^{-2}	3.07×10^{-3}	1.23×10^{-2}	3.60×10^{-2}
	t-value	0	1.188	3.707♯	5.939♯	2.962♯	4.498♯	5.523♯
f_5	Mean	**0**	**0**	4.85×10^{-10}	$5.62 \times 10^{+1}$	$6.59 \times 10^{+1}$	$2.84 \times 10^{+1}$	**0**
	Std	**0**	**0**	3.63×10^{-10}	$9.76 \times 10^{+0}$	$1.22 \times 10^{+1}$	$8.71 \times 10^{+0}$	**0**
	t-value	0	0	7.318♯	31.539♯	29.586♯	17.859♯	0
f_6	Mean	**0**	**0**	4.36×10^{-10}	$2.85 \times 10^{+1}$	$6.34 \times 10^{+1}$	$1.44 \times 10^{+1}$	1.00×10^{-1}
	Std	**0**	**0**	2.44×10^{-10}	$1.14 \times 10^{+1}$	$1.24 \times 10^{+1}$	$6.28 \times 10^{+0}$	3.16×10^{-1}
	t-value	0	0	9.787♯	13.693♯	28.005♯	12.559♯	1.733
f_7	Mean	**0**	**0**	3.45×10^{-7}	$4.10 \times 10^{+0}$	$9.60 \times 10^{+0}$	7.49×10^{-3}	7.82×10^{-15}
	Std	**0**	**0**	1.94×10^{-7}	$2.20 \times 10^{+0}$	$3.78 \times 10^{+0}$	1.14×10^{-2}	8.50×10^{-15}
	t-value	0	0	9.740♯	10.208♯	13.910♯	3.599♯	5.039♯
f_8	Mean	3.80×10^{-4}	$3.8210 - 4$	$\mathbf{1.27 \times 10^{-12}}$	$3.78 \times 10^{+3}$	$4.8410 + 3$	$3.6110 + 3$	$1.0810 + 3$
	Std	8.92×10^{-13}	$\mathbf{7.40 \times 10^{-13}}$	8.79×10^{-13}	$6.02 \times 10^{+2}$	$4.7610 + 2$	$3.0610 + 2$	$2.5910 + 2$
	t-value	0	$9.451 \times 10^{+6}$♯	$1.662 \times 10^{+9}$♯	34.392♯	55.693♯	64.617♯	22.839♯
f_9	Mean	**0**	3.60×10^{-7}	$3.46 \times 10^{+1}$	$7.13 \times 10^{+1}$	$7.07 \times 10^{+1}$	$4.44 \times 10^{+1}$	$1.01 \times 10^{+2}$
	Std	**0**	1.57×10^{-6}	$4.59 \times 10^{+0}$	$1.66 \times 10^{+1}$	$1.70 \times 10^{+1}$	$1.37 \times 10^{+1}$	$2.21 \times 10^{+1}$
	t-value	0	1.256	41.288♯	23.526♯	22.779♯	17.751♯	25.032♯
f_{10}	Mean	$\mathbf{8.88 \times 10^{-16}}$	4.03×10^{-12}	3.43×10^{-4}	$1.66 \times 10^{+1}$	2.94×10^{-1}	3.59×10^{-1}	$2.10 \times 10^{+0}$
	Std	$\mathbf{1.00 \times 10^{-31}}$	1.36×10^{-11}	1.91×10^{-4}	$1.10 \times 10^{+0}$	6.71×10^{-1}	5.93×10^{-1}	3.84×10^{-1}
	t-value	0	1.623	9.836♯	82.656♯	2.400♯	3.316♯	29.954♯
f_{11}	Mean	**0**	1.19×10^{-5}	7.04×10^{-10}	8.62×10^{-3}	1.48×10^{-3}	9.60×10^{-3}	5.54×10^{-2}
	Std	**0**	4.73×10^{-5}	1.25×10^{-11}	8.86×10^{-3}	3.12×10^{-3}	1.24×10^{-2}	3.97×10^{-2}
	t-value	0	1.378	308.477♯	5.329♯	2.598♯	4.240♯	7.643♯
f_{12}	Mean	$\mathbf{8.91 \times 10^{-5}}$	$2.84 \times 10^{+3}$	$1.70 \times 10^{+3}$	$3.57 \times 10^{+3}$	$5.60 \times 10^{+3}$	$3.78 \times 10^{+3}$	$3.64 \times 10^{+3}$
	Std	$\mathbf{1.64 \times 10^{-4}}$	$3.23 \times 10^{+2}$	$1.86 \times 10^{+2}$	$9.08 \times 10^{+2}$	$6.50 \times 10^{+2}$	$7.59 \times 10^{+2}$	$7.41 \times 10^{+2}$
	t-value	0	47.349♯	49.219♯	21.173♯	46.395♯	26.819♯	26.453♯
f_{13}	Mean	**0**	$4.24 \times 10^{+1}$	$3.77 \times 10^{+1}$	$7.88 \times 10^{+1}$	$7.74 \times 10^{+1}$	$4.36 \times 10^{+1}$	$8.80 \times 10^{+1}$
	Std	**0**	$9.84 \times 10^{+1}$	$5.56 \times 10^{+0}$	$1.88 \times 10^{+1}$	$1.40 \times 10^{+1}$	$8.96 \times 10^{+0}$	$2.59 \times 10^{+1}$
	t-value	0	2.360♯	37.139♯	22.958♯	30.281♯	26.653♯	18.610♯
f_{14}	Mean	**0**	1.15×10^{-8}	$3.07 \times 10^{+0}$	$8.48 \times 10^{+0}$	$1.85 \times 10^{+1}$	$2.50 \times 10^{+0}$	$1.43 \times 10^{+1}$
	Std	**0**	4.21×10^{-8}	$1.61 \times 10^{+0}$	$2.54 \times 10^{+0}$	$3.37 \times 10^{+0}$	$1.46 \times 10^{+0}$	$3.53 \times 10^{+0}$
	t-value	0	1.496	10.444♯	18.286♯	30.068♯	9.379♯	22.188♯

of the MMPSO are smaller than those of the PCLPSO on functions $f_1 - f_3$, f_8, f_9, and $f_{11} - f_{14}$, and the differences between the MMPSO and PCLPSO are statistically significant on these functions. Therefore, the optimization accuracy and reliability of the MMPSO are better than those of the PCLPSO on these functions. The mean and standard deviation values of the MMPSO are smaller than those of the CLPSO on functions $f_2 - f_4$, f_9, and $f_{11} - f_{14}$, and the differences between the MMPSO and CLPSO are statistically significant on these functions.

Table 6 Mean and standard deviation values of the fourteen benchmark functions for $100 - D$ problems

Function		MMPSO	PCLPSO	CLPSO
f_1	Mean	**0**	4.74×10^{-9}	1.39×10^{-4}
	Std	**0**	3.73×10^{-9}	3.69×10^{-5}
	t-value	0	6.960♯	20.632♯
f_2	Mean	$\mathbf{1.40 \times 10^{-6}}$	$7.29 \times 10^{+1}$	$9.48 \times 10^{+1}$
	Std	$\mathbf{2.82 \times 10^{-6}}$	$1.39 \times 10^{+1}$	$1.71 \times 10^{+1}$
	t-value	0	28.726♯	30.365♯
f_3	Mean	$\mathbf{8.88 \times 10^{-16}}$	1.08×10^{-5}	2.78×10^{-3}
	Std	$\mathbf{1.00 \times 10^{-31}}$	4.10×10^{-6}	6.26×10^{-4}
	t-value	0	14.428♯	24.324♯
f_4	Mean	**0**	1.73×10^{-7}	1.36×10^{-4}
	Std	**0**	5.88×10^{-7}	5.49×10^{-5}
	t-value	0	1.612	13.568♯
f_5	Mean	**0**	1.05×10^{-3}	$1.6 \times 10^{+1}$
	Std	**0**	1.72×10^{-3}	$4.16 \times 10^{+0}$
	t-value	0	3.344♯	2.107♯
f_6	Mean	**0**	$1.23 \times 10^{+1}$	$2.02 \times 10^{+0}$
	Std	**0**	$1.42 \times 10^{+1}$	$1.40 \times 10^{+0}$
	t-value	0	4.744♯	7.792♯
f_7	Mean	**0**	7.59×10^{-4}	$5.04 \times 10^{+0}$
	Std	**0**	2.86×10^{-4}	$1.52 \times 10^{+0}$
	t-value	0	14.536♯	18.161♯
f_8	Mean	1.30×10^{-3}	$\mathbf{1.270 \times 10^{-3}}$	$3.950 \times 10^{+1}$
	Std	$\mathbf{4.42 \times 10^{-5}}$	2.290×10^{-4}	$7.100 \times 10^{+1}$
	t-value	0	0.705	3.047♯
f_9	Mean	**0**	$4.85 \times 10^{+2}$	$4.93 \times 10^{+2}$
	Std	**0**	$9.11 \times 10^{+1}$	$9.22 \times 10^{+1}$
	t-value	0	29.160♯	29.287♯
f_{10}	Mean	8.88×10^{-16}	2.22×10^{-5}	2.71×10^{-3}
	Std	1.00×10^{-31}	1.17×10^{-5}	6.01×10^{-4}
	t-value	0	10.393♯	24.698♯
f_{11}	Mean	**0**	5.43×10^{-5}	3.43×10^{-3}
	Std	**0**	1.43×10^{-4}	1.66×10^{-3}
	t-value	0	2.080♯	11.317♯
f_{12}	Mean	4.25×10^{-5}	$1.84 \times 10^{+4}$	$1.84 \times 10^{+4}$
	Std	2.32×10^{-4}	$3.48 \times 10^{+3}$	$3.37 \times 10^{+3}$
	t-value	0	28.960♯	29.905♯
f_{13}	Mean	**0**	$4.42 \times 10^{+2}$	$3.79 \times 10^{+2}$
	Std	**0**	$8.38 \times 10^{+1}$	$7.42 \times 10^{+1}$
	t-value	0	28.889♯	27.977♯
f_{14}	Mean	**0**	7.16×10^{-1}	$4.39 \times 10^{+0}$
	Std	**0**	8.53×10^{-1}	$1.30 \times 10^{+0}$
	t-value	0	4.598♯	18.496♯

Therefore, the optimization accuracy and reliability of the MMPSO are better than those of the CLPSO on these functions. The mean and standard deviation values of the MMPSO are smaller than those of the PSO-CF and UPSO on functions $f_2 - f_{14}$, and the differences between the MMPSO and PSO-CF and UPSO are statistically significant on these functions. Therefore, the optimization precision and reliability of the MMPSO are better than those of the PSO-CF and UPSO on these functions. The mean and standard deviation values of the MMPSO are smaller than those of the FDR-PSO on functions $f_2 - f_9$ and $f_{11} - f_{14}$, and the differences between the MMPSO and FDR-PSO



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

(g) f_7 (h) f_8 (i) f_9

(j) f_{10} (k) f_{11} (l) f_{12}

(m) f_{13} (n) f_{14}

Fig. 4 Convergence characteristics of MMPSO

are statistically significant on these functions. MMPSO and CPSO-H have the same optimization performance on function f_5, but the performance of the CPSO-H is worse than that of the MMPSO on other thirteen functions. The results of the comparisons show that the MMPSO outperforms the PCLPSO, CLPSO, PSO-CF, UPSO, FDR-PSO, and CPSO-H when searching for 10-D problem space.

Table 5 shows the mean and standard deviation values of the fourteen benchmark functions for $30-D$ problems. The differences between the MMPSO and PCLPSO on functions f_1-f_3, f_8, f_{12}, and f_{13} are statistically significant. Meanwhile, the mean and standard deviation values of the MMPSO are smaller than PCLPSO on these functions. CLPSO has much better results than MMPSO only on function f_3 and f_8, but the MMPSO performs better than CLPSO on functions f_1, f_2, f_4-f_7, and f_9-f_{14}. The differences between the MMPSO and PSO-CF are statistically significant when optimizing all functions, so the optimization accuracy and reliability of the MMPSO are better than PSO-CF. The optimization accuracy and reliability of the MMPSO are better than UPSO on all functions except on function f_3. The differences between the MMPSO and FDR-PSO are statistically significant when optimizing all functions except on function f_1, so the optimization accuracy and reliability of the MMPSO are better than FDR-PSO. Although the CPSO-H and the MMPSO have the same performance on function f_5, and the performance of them are close on function f_6. The optimization results of the CPSO-H are worse than MMPSO on other twelve functions. The optimization results demonstrate that the MMPSO also outperforms the PCLPSO, CLPSO, PSO-CF, UPSO, FDR-PSO, and CPSO-H when searching for $30-D$ problem space.

In order to demonstrate the optimization performance of the MMPSO for the high-dimensional problems, comparisons of $100-D$ problems are presented in the following. Table 6 shows the mean and standard deviation values of the fourteen benchmark functions for $100-D$ problems. Only the performance results of the PCLPSO and CLPSO for $100-D$ problem are available in [23]. As shown in Table 6, the MMPSO has smaller mean and standard deviation values than PCLPSO on all functions except on function f_8. The differences between the MMPSO and PCLPSO on functions f_4 and f_8 are not statistically significant, so their performances are close when optimizing these functions. MMPSO has smaller mean and standard deviation values than the CLPSO on all functions, and the differences between the MMPSO and CLPSO are statistically significant on all functions. When searching for $100-D$ problem space, the optimization results validate the performance of the MMPSO is better than PCLPSO and CLPSO.

It can be observed from the Tables 4 and 5 that the optimization results of the PSO-CF, UPSO, FDR-PSO, and CPSO-H become worse on most of functions when the problem dimension increases from 10 to 30. As can be seen from Tables 4, 5, and 6, the optimization performances of the PCLPSO and CLPSO become worse on many functions when the problem dimension increases from 10 to 30. And their optimization performances are extremely worse for many high-dimensional problems ($D = 100$ problems). However, the performances of the MMPSO for most $100-D$ problems are almost close to those with $D = 30$ and $D = 10$ problems. Benefitting from the strong coordinate optimization ability of the multi-swarm with multi-velocity, the optimization precision and the optimization reliability of the MMPSO are better than the PSO-CF, UPSO, FDR-PSO, CPSO-H, PCLPSO, and CLPSO

The convergence characteristics in terms of the best fitness value of the MMPSO for all functions with different dimensions are shown in Fig. 4. As can be seen from Fig. 4, even for the high-dimensional problems, the MMPSO can also quickly find the global optimum area. Meanwhile, the MMPSO still has high optimization accuracy for the high-dimensional problems.

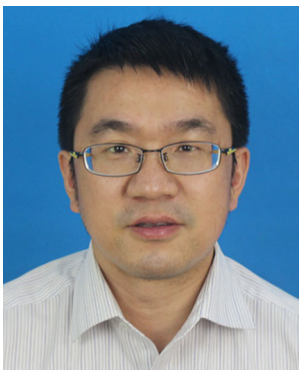
4 Conclusions

In this paper, the MMPSO is proposed to improve the optimization performance of the traditional PSO algorithm when searching for high-dimensional complex problems. It comprises three particle swarms and three velocity update methods. The main particle swarm with one velocity update method is the original particle swarm, and it constructs the global auxiliary particle swarm and local auxiliary particle swarm. The global auxiliary particle swarm with two different velocity update methods is designed to enhance the global optimization ability of the algorithm. And the local auxiliary particle swarm is designed to enhance the local optimization ability of the algorithm. Therefore, their combination can greatly improve the global optimization performance and local optimization performance of the algorithm.

Fourteen benchmark functions are selected to test and compare the performance of the MMPSO and the other eight improved PSO algorithms. Comprehensive experimental test results show that the optimization precision and reliability of the MMPSO are better than those of the other various improved PSO algorithms, such as the PCLPSO, CLPSO, PSO-CF, and so on when the dimension of the problem space increases. Even for the high-dimensional problems, the optimization performance of the MMPSO is still better than that of the PCLPSO and CLPSO, making it attractive to the applications in the energy management system, control scheme of the distributed generation units, and other research fields.

References

1. Abualigah LM, Khader AT, Al-Betar MA, Alomari OA (2017) Text feature selection with a robust weight scheme and dynamic dimension reduction to text document clustering. Expert Syst Appl 84:24–36
2. Alswaitti M, Albughdadi M, Isa NAM (2018) Density-based particle swarm optimization algorithm for data clustering. Expert Syst Appl 91:170–186
3. Bamakan SMH, Wang H, Yingjie T, Shi Y (2016) An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization. Neurocomputing 199:90–102
4. Chang WD (2017) Multimodal function optimizations with multiple maximums and multiple minimums using an improved PSO algorithm. Appl Soft Comput 60:60–72
5. Chen J, Zheng J, Wu P, Zhang L, Wu Q (2017) Dynamic particle swarm optimizer with escaping prey for solving constrained non-convex and piecewise optimization problems. Expert Syst Appl 86:208–223
6. Gülcü S, Kodaz H (2015) A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization. Eng Appl Artif Intell 45:33–45
7. Gunasundari S, Janakiraman S, Meenambal S (2016) Velocity bounded boolean particle swarm optimization for improved feature selection in liver and kidney disease diagnosis. Expert Syst Appl 56:28–47
8. Kadirkamanathan V, Selvarajah K, Fleming PJ (2006) Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Trans Evol Comput 10(3):245–255
9. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE international conference on neural networks proceedings, vol 4, pp 1942–1948
10. Kermadi M, Berkouk EM (2017) Artificial intelligence-based maximum power point tracking controllers for photovoltaic systems: comparative study. Renew Sust Energ Rev 69:369–386
11. Khan SU, Yang S, Wang L, Liu L (2016) A modified particle swarm optimization algorithm for global optimizations of inverse problems. IEEE Trans Magn 52(3):1–4
12. Kiranyaz S, Pulkkinen J, Gabbouj M (2011) Multi-dimensional particle swarm optimization in dynamic environments. Expert Syst Appl 38(3):2212–2223
13. Kumar EV, Raaja GS, Jerome J (2016) Adaptive PSO for optimal LQR tracking control of 2 dof laboratory helicopter. Appl Soft Comput 41:77–90
14. Li NJ, Wang WJ, Hsu CCJ, Chang W, Chou HG, Chang JW (2014) Enhanced particle swarm optimizer incorporating a weighted particle. Neurocomputing 124:218–227
15. Li XM, Sun YL, Chen WN, Zhang J (2017) Multi-swarm particle swarm optimization for payment scheduling. In: 2017 seventh international conference on information science and technology (ICIST), pp 284–291
16. Liu R, Li J, fan J, Mu C, Jiao L (2017) A coevolutionary technique based on multi-swarm particle swarm optimization for dynamic multi-objective optimization. Eur J Oper Res 261(3):1028–1051
17. Liu ZG, Ji XH, Liu YX (2018) Hybrid non-parametric particle swarm optimization and its stability analysis. Expert Syst Appl 92:256–275
18. Liu ZH, Wei HL, Zhong QC, Liu K, Li XH (2017) GPU implementation of DPSO-RE algorithm for parameters identification of surface PMSM considering VSI nonlinearity. IEEE J Emerg Select Topics Power Electron 5(3):1334–1345
19. Ma K, Hu S, Yang J, Xu X, Guan X (2017) Appliances scheduling via cooperative multi-swarm PSO under day-ahead prices and photovoltaic generation. Appl Soft Comput 62:504–513
20. Moradi MH, Bahrami FV, Mohammad A (2017) Power flow analysis in islanded micro-grids via modeling different operational modes of DGs: a review and a new approach. Renew Sust Energ Rev 69:248–262
21. Nieto PG, García-Gonzalo E, Fernández JA, Muñiz CD (2016) A hybrid PSO optimized SVM-based model for predicting a successful growth cycle of the spirulina platensis from raceway experiments data. J Comput Appl Math 291:293–303
22. Pandit M, Srivastava L, Sharma M (2015) Performance comparison of enhanced PSO and DE variants for dynamic energy/reserve scheduling in multi-zone electricity market. Appl Soft Comput 37:619–631
23. Rahmani M, Ghanbari A, Ettefagh MM (2016) Robust adaptive control of a bio-inspired robot manipulator using bat algorithm. Expert Syst Appl 56:164–176
24. Samal NR, Konar A, Nagar A (2008) Stability analysis and parameter selection of a particle swarm optimizer in a dynamic environment. In: 2008 second UKSIM European symposium on computer modeling and simulation, pp 21–27
25. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: IEEE international conference on evolutionary computation, pp 69–73
26. Shirani H, Habibi M, Besalatpour A, Esfandiarpour I (2015) Determining the features influencing physical quality of calcareous soils in a semiarid region of Iran using a hybrid PSO-DT algorithm. Geoderma 259-260:1–11
27. Tanweer M, Suresh S, Sundararajan N (2015) Self regulating particle swarm optimization algorithm. Inf Sci 294:182–202
28. fang Wang Z, Wang J, mei Sui Q, Jia L (2017) The simultaneous measurement of temperature and mean strain based on the distorted spectra of half-encapsulated fiber bragg gratings using improved particle swarm optimization. Opt Commun 392:153–161
29. Xu G (2013) An adaptive parameter tuning of particle swarm optimization algorithm. Appl Math Comput 219(9):4560–4569
30. Yang C, Gao W, Liu N, Song C (2015) Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight. Appl Soft Comput 29:386–394
31. Yang G, Zhou F, Ma Y, Yu Z, Zhang Y, He J (2018) Identifying lightning channel-base current function parameters by powell particle swarm optimization method. IEEE Trans Electromagn Compat 60(1):182–187
32. Yuan Q, Yin G (2015) Analyzing convergence and rates of convergence of particle swarm optimization algorithms using stochastic approximation methods. IEEE Trans Autom Control 60(7):1760–1773



Yong Ning was born in Hunan Province, China, in 1980. He received the B.S. and M.S. degrees from Hunan University, Changsha, China, in 2003 and 2010, where he is currently working toward the Ph.D. degree. His current research interests include power electronics system control and application, modeling and control of micro grid and intelligence algorithm.

Zishun Peng received the B.S. degree from Hunan Institute of Engineering, Xiangtan, China, in 2009, in electric information engineering. He received the M.S. degree from Wenzhou University, Wenzhou, China, in 2013, in computer applications technology. Mr. Peng is currently pursuing the Ph.D. degree in Electric Engineering at the College of Electrical and Information Engineering, Hunan University, Changsha, China. His research interests include intelligence algorithm, power electronics devices and their applications.

Daqiang Bi was born in Jilin Province, China, in 1973. He received his MSc degree in Electrical Engineering from Shenyang University of Technology in 1999. He was awarded his Ph.D. by Tsinghua University in 2003. He worked as a post-doctoral researcher at Tsinghua University from Aug. 2003 to Jun. 2005. Currently, he is a senior engineer in State Key Laboratory of Power Systems and the director of Laboratory on Power Electronics and Electric Machine Control, Dept. of Electrical Engineering, Tsinghua University, his research being mainly power system relay protection, application of power electronics to power system and intelligence algorithm.

Yuxing Dai was born in China in 1956. He received Ph.D. degree from Central South University, China. He is currently a professor with College of Electrical and Information Engineering, Hunan University, China. His research interests include intelligence algorithm, power electronics technology, converter technology and application.

Jun Wang received the B.S. degree at Huazhong University of Science and Technology, Wuhan, China in 2000. He received the M.S. degree in Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China in 2003. He received the M.E. degree at University of South Carolina, Columbia, SC, USA in 2005, and the Ph.D. degree at North Carolina State University, Raleigh, NC, USA in 2010 respectively, all in Electrical Engineering. He then worked as a device design engineer at Texas Instruments, Inc. in Bethlehem, PA, USA between 2010 and 2013. He became a professor in the College of Electrical and Information Engineering at Hunan University in 2014. He has authored and coauthored more than 30 journal and conference publications and holds 3 issued U.S. patents. His research interests include power semiconductor devices and intelligence algorithm.