Ain Shams Engineering Journal 10 (2019) 847–866

Contents lists available at ScienceDirect

Ain Shams Engineering Journal

journal homepage: www.sciencedirect.com

Engineering Physics and Mathematics

A modified particle swarm optimization algorithm based on velocity updating mechanism

Chunfeng Wang *, Wenxin Song

College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, PR China

ARTICLE INFO

Article history:
Received 12 October 2018
Revised 6 January 2019
Accepted 5 February 2019
Available online 9 March 2019

Keywords:
Probability selection
Particle swarm optimization
Convex combination
Opposite learning mechanism

ABSTRACT

Due to less parameters and simple operations in PSO, PSO has attracted the attention of many researchers. However, it amy fall into local optimum and the search precision is not high. Therefore, this paper introduces an improved PSO (CNPSO). There are two new formulas: (1) when the individual is not the best particle, the *gbest* is replaced by a particle, which is selected from a set based on the probability calculation. Such mechanism can help the algorithm to escape local position. (2) when the individual is the best particle, it is combined with a randomly selected particle to generate a convex combination, after that opposite learning is adopted to get a reverse solution. This operation can maintain the diversity of population. Finally, CNPSO is compared with several algorithms in three experiments, and used to optimize the spring design problem. The results indicate CNPSO has good performance and high search precision.

1. Introduction

Meta-heuristic algorithm is a combination of random algorithms and local search algorithms. For the features of simple structure and easy realization, it has received a lot of attention. Up to now, many meta-heuristic algorithms have been proposed, such as Simulated Annealing Algorithm [1], Firefly Algorithm [2], Particle Swarm Optimization [3], Cuckoo Search [4], Artificial Fish School Algorithm [5]. Besides, it plays an important role in solving many real problems, such as flexible job shop scheduling problems [6], transmission network planning issues [7] and so on.

Particle Swarm Optimization (PSO) was proposed by Kennedy and Eberhart through simulating the foraging behavior of birds. Although it is easy to operate and has a fast convergence speed, it performs poorly in balancing the exploration and exploitation, and the accuracy of algorithm is not high.

To enhance the performance of PSO, many people have done a lot of improvements. Olivas et al. [8–10] introduced fuzzy logic system into PSO to dynamically adjust the parameters, which can improve the performance of PSO. He et al. [11] proposed a damping factor α, and a cooperative mechanism between the global-best-oriented and the local-best-oriented swarms to make algorithm find the global optima more quickly. To enhance the search ability of algorithm, Chen et al. [12] used two different crossover operations to breed promising exemplars to guide particles. Tsai [13] added the unification factor to algorithm to balance the cognitive learning and social learning of the velocity update formula. To increase the diversity of the population, Pan et al. [14] introduced three strategies based on elite and Ye et al. [15] divided the population into multi-subgroups. Simon et al. [16] chose the search direction of particles based on the worst individuals instead of the best individuals to improve the algorithm. Yang et al. [17] proposed a dynamic topology strategy to enhance the capability of PSO. By combining cellular automata and basic PSO, Liu et al. [18] proposed two versions of CPSO to improve the search ability. Patra et al. [19] put forward an orthogonal PSO to avoid happening collision phenomenon in search process. In order to prevent the algorithm falling into local optimum, a combination of Levy flight and random learning strategy was introduced in [20], a cellur population structure was added in [21], an evolution strategy was applied in [22] and a local optima topology (LOT) structure based on the comprehensive learning particle swarm optimizer is designed in [23].

* Corresponding author.
E-mail address: wangchunfeng09@126.com (C. Wang).

Peer review under responsibility of Ain Shams University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.asej.2019.02.006
2090-4479/© 2019 The Authors. Published by Elsevier B.V. on behalf of Faculty of Engineering, Ain Shams University.

Because the setting of parameters has a great influence on the algorithm, many scholars have done a lot of research. Xu [24] used the information of average absolute value of velocity to adjust parameters to find a high quality solution. To balance the local and global search of algorithm, Liu et al. [25] proposed a hybrid non-parametric algorithm, which incorporates two crossover operations and canonical learning mechanism. Feng et al. [26] adopted chaotic map to defined w in the velocity formula to enhance the ability of algorithm. Some other adjustments for parameters can be found in [27–31].

Besides, there are many hybrid algorithms about PSO to solve optimization problems or improve the performance of algorithm. Liu et al. [32] combined PSO with DE algorithm to solve the numerical and engineering optimization problems. Ali et al. [33] combined PSO with GA to solve large scale optimization functions. Nazari et al. [34] combined PSO with SAA to obtain a high quality solution. Xia et al. [35] combined PSO with FA to improved the search ability. Yadav et al. [36] added the gravitational search to PSO to solve some constraint problems.

To prevent premature convergence of algorithm and improve the precision of solution, a modified particle swarm optimization algorithm based on velocity update mechanism is introduced in this paper. Two strategies are proposed to improve search ability and the diversity maintain ability. (1) For the individual is not the best particle, it is guided by a particle with high quality instead of global best particle (*gbest*), and the selection operation is depended on probability calculation, which can make particle learn good information from other high quality particle rather than *gbest* to avoid falling into local optimum. (2) For the best individual, a particle is selected randomly from the population, then the convex combination of best particle and randomly selected particle is generated. After that, opposite learning strategy is conducted on it to increase the widespread distribution of solutions and maintain the diversity of population.

The main contributions in this paper includes:

(1) The best particle and other particles in population are updated in different ways, which is not distinguished in basic PSO.
(2) The *gbest* in velocity formula is replaced by a high quality solution to avoid falling into a local optimum.
(3) The property of convex combination and the opposite learning strategy are used in the update formula of the best particle, which can maintain the diversity of population and improve the global search ability.

This article is organized as follows: Section 2 gives a brief introduction of the basic PSO. The new improved algorithm (CNPSO) is described in Section 3. The computational complexity of the proposed method is analyzed in Section 4. The comparative experiments are presented in Section 5. Section 6 presents a summary and future work about the proposed algorithm.

2. Basic knowledge of PSO algorithm

2.1. Basic PSO algorithm

In nature, the movement of individuals in flocks is independent, but they exhibit similar synchronicity throughout the flight. The conscious concentration of such discrete groups has been studied by many scholars. Through simulating the foraging behavior of flocks, Kennedy and Eberhart proposed a particle swarm optimization (PSO) algorithm.

Considering N-dimensional search space, the number of particles is N_p, the position of particle i is $x_i = [x_i^1, x_i^2, \ldots, x_i^N]$, and the velocity vector is $v_i = [v_i^1, v_i^2, \ldots, v_i^N]$. Meanwhile, the historical best position of particle i is $pbest_i = [pbest_i^1, pbest_i^2, \ldots, pbest_i^N]$, and the global best position is $gbest = [gbest^1, gbest^2, \ldots, gbest^N]$.

The position and velocity of particle i in iteration t are updated as follows:

$$v_i^d(t+1) = v_i^d(t) + c_1 r_1 (pbest_i^d(t) - x_i^d(t)) + c_2 r_2 (gbest^d(t) - x_i^d(t)) \quad (1)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (2)$$

In (1), the second part is the "cognitive part", which reflects the particle's memory about its own historical experience; the third part is the "social part", which reacts the information sharing and cooperation among particles. c_1 and c_2 are learning factors. If $c_1 = 0$, there is no cognitive part but only the social part, each particle is only influenced by the historical optimal information of the whole population when it moves. Similarly, if $c_2 = 0$, it means particle's movement is influenced by its own historical optimal information without information exchange. If $c_1 = c_2 = 0$, the particle will move toward the original direction.

To improve PSO algorithm, many scholars added inertia weight w to the first part of the velocity updating formula, and the general formula is:

$$v_i^d(t+1) = w v_i^d(t) + c_1 r_1 (pbest_i^d - x_i^d) + c_2 r_2 (gbest^d - x_i^d) \quad (3)$$

2.2. Process of basic PSO

Step 1: Initialize the population: suppose that the number of the population is *Np*, and the dimension of space is *N*. Position and speed of particles are generated randomly.
Step 2: Calculate the fitness value of particles in the population.
Step 3: The fitness value of each particle is compared with their own historical best position (*pbest*), and *pbest* will be replaced if the former is better than it.
Step 4: The fitness value of each particle is compared with the global best position of the population (*gbest*), and *gbest* will be replaced if the former is better than it.
Step 5: According to (2) and (3), the position and velocity of particles are updated by dimensions.
Step 6: If the algorithm does not reach the maximum number of iterations, then return to Step 2 and continue.

3. The improved PSO:CNPSO

3.1. Motivation

In the basic PSO, each particle update their position by moving towards the particle's personal best solution (pbest) and the global best solution (gbest). Although the information of good position can be used in the update formula, it may cause particles fall into local optimum, and the precision of obtained solution is not high. Considering the shortcomings of the basic PSO, a novel particle swarm algorithm is proposed in this paper to improve the performance of basic PSO.

In order to avoid premature convergence, the global best particle in the velocity update is replaced by a high quality solution, which is selected based on probability calculation. This can make each particle obtain information from other better individuals, not just limited by one particle, to help algorithm escape local position efficiently. Besides, the best particle is updated differently with others in this paper. The convex combination between best particle and a randomly selected particle form population is given to generate a new position, then the opposition learning strategy is utilized to get a reverse solution. This operation can make algorithm search the space more widely and main the diversity of population. In this paper, two update operations for the best particle and other particles in population can help algorithm escape local optimum and maintain diversity of population. More details are presented in follow subsections.

3.2. Two update formulas

In this subsection, two update formulas are introduced in detail. The best particle and other particles in the population will be updated in different ways. If the particle is not the best one, it will be guided by the historical best position (*pbest*) and a particle that selected from a set in the way of probability. So each particle is leaded by a better one instead of *gbest*, which can avoid premature convergence. If the particle is the best one, a new position is generated by a convex combination of the best particle and a randomly selected particle from the population, and then the opposition learning strategy is conducted on it to get a new position again. This operation can make algorithm has more opportunity to search different space, thus enhancing the diversity of search.

Considering two cases:

(1) x_i is not the best individual. To guarantee each particle can guided by more other good particles, a probability calculation is used to select a particle with high quality randomly. And the calculation method depends on the fitness function, which has the characteristics that the smaller the function value, the bigger the fitness value. First, comparing the fitness value of x_i with other particles in the population by (5). All particles whose fitness value is better than x_i is put into a set A. Second, the probability value $Prob(k)$ of each particle $x_k \in A$ is calculated based on the fitness function. Then, a comparison between $Prob(k)$ and a random number r is conducted. If $Prob(k)$ is bigger than r, the kth particle in set A will be selected to guide the moving direction of x_i. The corresponding formulas are as follows

$$Prob(k) = \frac{fit(x_k)}{\sum_{x_t \in A} fit(x_t)} \tag{4}$$

where $fit(x)$ is defined as follows:

$$fit(x) = \begin{cases} \frac{1}{1+f(x)}, & \text{if } f(x) \geqslant 0, \\ 1 + abs(f(x)), & \text{else.} \end{cases} \tag{5}$$

From the above analysis, it can be seen that the particle with high quality is selected randomly rather than the best one in the set. This is because the randomness of select enables each particle to learn from more high quality solution, thus helping algorithm escape local optimum efficiently.

By (4) and (5), we select a leader $x_j \in A$, and replace *gbest* with x_j. Finally, the new velocity equation is given as follows

$$v_i^d(t+1) = wv_i^d(t) + c_1 r_1 (pbest_i^d - x_i^d(t)) + c_2 r_2 (x_j^d(t) - x_i^d(t)) \tag{6}$$

In this paper, the value of w is changed from large to small, and the function is defined as follow

$$w = w_{max} - \frac{w_{max} - w_{min}}{Itmax} iter \tag{7}$$

where w_{max}, w_{min} are the maximum weight value and minimum weight value respectively, *iter* is the number of current iterations and *Itmax* is maximum number of iterations.

Based on above analysis, the position of x_i is updated by the following formula

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{8}$$

(2) x_i itself is the best particle of the population. Because the opposition learning strategy can generate reverse solution to increase the diversity of population, it is used to update the position of x_i in this paper. Besides, to search the space more widely, the opposition learning is not conducted on the best particle directly, but on a convex combination of the best particle and another particle in the population. First, a particle x_k is selected randomly from the population, and then x_i and x_k is combined through the way of convex combination to get $\hat{x}_i$ (see (9)). Finally, the opposition learning strategy is adopted on $\hat{x}_i$ to generate a new solution (see (10)). As the coefficient of convex combination is $[0,1]$, which can generate different new solutions. So the combination of this two strategies can make the algorithm search the space as much as possible to find a potential solution. The update formulas are as follows

$$\hat{x}_i = (1-\lambda)x_i(t) + \lambda x_k(t) \tag{9}$$

$$x_i^{new} = x_{min} + x_{max} - \hat{x}_i \tag{10}$$

where λ belongs to $[0,1]$.

To illustrate this case more clearly, Fig. 1 gives the corresponding movement of x_i. In Fig. 1, x^* denotes the optimal solution, $\hat{x}_i$ denotes the convex combination of x_i and x_k, and x_i^{new} denotes the new solution obtained by the opposition learning. From the moving trajectory of x_i, it can be seen that the new position is closer to the optimal solution x^* after adopting the convex combination and opposition learning strategy on x_i. It means the opportunity of finding best solution is increased, so this two operations can enhance the diversity of search and improve the global search ability.

According to the description of CNPSO algorithm above, the pseudo-code is given in Algorithm 1.

Algorithm 1. Pseudo-code of CNPSO

```
01: initialize the population size $Np$, the dimension of the
      space $N$, $c_1$, $c_2$, the maximum number of
    iterations $Itmax$, $v_{max}$, $v_{min}$.
02: Set $pbest_i = x_i (i = 1, 2, \ldots, Np)$, calculate the fitness of $x_i$
      and find $gbest$.
03: While $iter < Itmax$ do
04: Velocity and position of each particle are generated
      randomly.
05: Calculate the fitness of the whole population by (5)
06: For $i = 1$ to $N_p$ do
07:     Determine the set $A$
08: If $x_i$ is not the best solution
09:     Generate $x_i$ by (6), (7) and (8),
10: Else
11:     Generate $x_i$ by (9) and (10),
12: End
13: If $f(x_i) < f(pbest_i)$
14:     $pbest_i = x_i$
15:     End
16: If $f(x_i) < f(gbest)$
17:     $gbest = x_i$
18:     End
19: $iter = iter + 1$
20: End
```

4. Experiments comparison and analysis

In order to verify the performance of CNPSO algorithm, we conduct four experiments to compare CNPSO with other algorithms. In

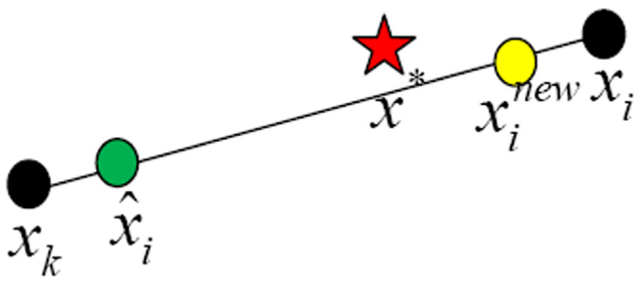

Fig. 1. The movement of x_i in case 2.

experiment 1, the comparison of CNPSO and other five PSOs is done to test the quality of CNPSO. To further verify the performance of the proposed algorithm, CNPSO is compared with some improved ABC algorithms and some DEs, CAs in experiment 2 and experiment 3 respectively. In experiment 4, CNPSO and other algorithms are used to optimize a practical problem to test the capability of proposed algorithm.

4.1. Experiment 1: Compared with five PSOs

In this subsection, CNPSO is compared with basic PSO [3] and other four improved algorithms: AIWPSO [37], PSOGSA [38], PSOd [39] and H-PSO-SCAC [40] on 27 test functions. The minimum value(min), mean value (mean) and standard deviation (std) of the obtained results is used as the performance testing indicator. Besides, all algorithms are run in Matlab 9.2.0 (Win 64) of personal computer with Inter(R) Core(TM) i5-4258U CPU @2.40 GHz under Windows 7 system.

4.1.1. Benchmark function

In experiment 1, 27 test functions are employed, which include unimodal functions, multi-modal functions and rotational functions. f_1-f_{12} are unimodal functions, $f_{13}-f_{24}$ are multi-modal functions, $f_{25}-f_{27}$ are multi-modal rotational functions. These functions are tested on 30 and 100 dimensions respectively. The range of variable and the optimal value of each function are given in Table 1.

Table 1
22 benchmark functions in experiment 1.

Functions	Range	Optimal value
$f_1=0.26(x_1^2+x_2^2)-0.48x_1x_2$	$[-10,10]$	0
$f_2=4x_1^2-2.1x_1^4+(x_1^6)/3+x_1x_2-4x_2^2+4x_2^4$	$[-5,5]$	-1.0316
$f_3=\sum_{i=1}^D x_i^2$	$[-100,100]$	0
$f_4=\sum_{i=1}^D \lvert x_i\rvert+\prod_{i=1}^D \lvert x_i\rvert$	$[-10,10]$	0
$f_5=\sum_{i=1}^D(\sum_{j=1}^i x_j)^2$	$[-100,100]$	0
$f_6=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant D\}$	$[-100,100]$	0
$f_7=\sum_{i=1}^D ix_i^2$	$[-10,10]$	0
$f_8=\sum_{i=1}^D ix_i^4$	$[-1.28,1.28]$	0
$f_9=\sum_{i=1}^D \lvert x_i\rvert^{(i+1)}$	$[-1,1]$	0
$f_{10}=\sum_{i=1}^D(10^6)^{\frac{i-1}{D-1}}x_i^2$	$[-100,100]$	0
$f_{11}=\sum_{i=1}^D(\lfloor x_i+0.5\rfloor)^2$	$[-1.28,1.28]$	0
$f_{12}=\sum_{i=1}^D ix_i^4+random[0,1)$	$[-1.28,1.28]$	0
$f_{13}=\sum_{i=1}^D(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]$	0
$f_{14}=-20\exp(-0.2*\sqrt{\sum_{i=1}^D x_i^2/D})-\exp(\sum_{i=1}^D\cos(2\pi x_i/D)+20+e$	$[-32,32]$	0
$f_{15}=\frac{1}{4000}\sum_{i=1}^D x_i^2-\prod_{i=1}^n cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]$	0
$f_{16}=0.5+\frac{sin(\sqrt{\sum_{i=1}^D x_i^2})^2-0.5}{(1+0.001\sum_{i=1}^D x_i^2)^2}$	$[-100,100]$	0
$f_{17}=\frac{\sum_{i=1}^D(x_i^4-16x_i^2+5x_i)}{D}$	$[-5,5]$	-78.3323
$f_{18}=\sum_{i=1}^D \lvert x_i sin(x_i)+0.1x_i\rvert$	$[-10,10]$	0
$f_{19}=\begin{cases}\sum_{i=1}^D(x_i^2-10\cos(2\Pi x_i)+10), & \lvert x_i\rvert<0.5\\ \sum_{i=1}^D((\frac{random(2x_i)}{2})^2-10\cos(\Pi random(2x_i))+10), & \lvert x_i\rvert\geqslant 0.5\end{cases}$	$[-5.12,5.12]$	0
$f_{20}=\frac{\Pi}{D}10\sin^2(\Pi y_1)+\frac{\Pi}{D}\sum_{i=1}^{D-1}(y_i-1)^2[1+10sin^2(\Pi y_{i+1})]$		
$+\frac{\Pi}{D}(y_D-1)^2+\sum_{i=1}^D u(x_i,10,100,4)$		
$y_i=1+\frac{x_i+1}{4}$		
$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	$[-50,50]$	0
$f_{21}=418.98288727243369*D-\sum_{i=1}^D x_i sin(\sqrt{\lvert x_i\rvert})$	$[-500,500]$	0
$f_{22}=\sum_{i=1}^D(10^{\frac{i-1}{D-1}}x_i)^2-10\cos(2\Pi 10^{\frac{i}{D-1}}x_i)+10$	$[-5.12,5.12]$	0
$f_{23}=0.1\{\sin^2(3\Pi x_1)+\sum_{i=1}^D(x_i-1)^2[1+sin^2(3\Pi x_i+1)]\}$		
$+0.1\{(x_D-1)^2[1+\sin^2(2\pi x_D)]\}+\sum_{i=1}^D u(x_i,5,100,4)$		
$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	$[-50,50]$	0
$f_{24}=\sin(\pi\omega_1)^2+\sum_{i=1}^{D-1}(w_i-1)^2[1+10\sin(\pi\omega_i+1)^2]$		
$+(\omega_D-1)^2[1+\sin(2\pi\omega_D)^2],\omega_i=1+\frac{x_i-1}{4}$	$[-10,10]$	0
$f_{25}=\sum_{i=1}^D(20^{\frac{i-1}{D-1}}*z_i)^2,z=x*M$	$[-100,100]$	0
$f_{26}=(1000x_1)^2+\sum_{i=2}^D z_i^2,z=x*M$	$[-100,100]$	0
$f_{27}=\frac{1}{4000}\sum_{i=1}^D z_i^2-\prod_{i=1}^n cos(\frac{z_i}{\sqrt{i}})+1,z=x*M$	$[-600,600]$	0

Table 2
Parameter configurations of the comparison algorithms.

Algorithm	Parameters setting
PSO	$c_1 = 2, c_2 = 2, w_max = 0.9, w_min = 0.1$
AIWPSO	$c_1 = 1.2, c_2 = 1.2, w_{max} = 0.9, w_{min} = 0.4$
PSOGSA	$c_1 = 0.5, c_2 = 1.5, G_0 = 1, \alpha = 20, w = rand$
CNPSO	$c_1 = 1.2, c_2 = 1.2, w_{max} = 0.85, w_min = 0.4$
H-PSO-SCAC	$\partial = 2, \delta = 0.5$

4.1.2. Parameter test

The parameter plays an important role in improving the algorithm, and an appropriate parameters can improve the performance of the algorithm. Therefore, an experiment is done to analyze the influence of two learning factor (c_1, c_2) with 10 different values on the improved algorithm. The average and standard deviation of obtained solutions are used to evaluate the performance of CNPSO. And the results are presented in Tables 3 and 4. According to the data given in Tables, we can see the results

Table 3
Results obtained by CNPSO with different values of c_1 and c_2 on 30-dimension.

Fun.	$c_1 = c_2 = 1.0$	$c_1 = c_2 = 1.3$	$c_1 = c_2 = 1.5$	$c_1 = c_2 = 1.8$	$c_1 = c_2 = 2.0$
f_1	0(0)	0(0)	0(0)	0(0)	0(0)
f_2	−1.0316(1.6231e−11)	−1.0316(9.8465e−13)	−1.0316(1.4709e−10)	−1.0316(1.0169e−06)	−1.0316(7.8732e−06)
f_3	0(0)	0(0)	0(0)	0(0)	(0)
f_4	0(0)	0(0)	0(0)	0(0)	(0)
f_5	0(0)	0(0)	0(0)	0(0)	(0)
f_6	0(0)	0(0)	0(0)	0(0)	(0)
f_7	0(0)	0(0)	0(0)	0(0)	(0)
f_8	0(0)	0(0)	0(0)	0(0)	(0)
f_9	0(0)	0(0)	0(0)	0(5.7386e−284)	2.2396e−63(1.1914e−62)
f_{10}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{11}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{12}	8.3354e−04(5.1098e−05)	3.1951e−04(1.6751e−04)	8.5000e−03(4.4837e−04)	2.4864e−03(1.3112e−03)	5.2000e−03(2.1534e−03)
f_{13}	0(0)	0(0)	0(0)	1.0117e−09(5.5415e−09)	1.2873e+01(3.0145e+01)
f_{14}	−8.8818e−16(0)	−8.8818e−16(0)	−8.8818e−16(0)	−8.8818e−16(0)	−8.8818e−16(0)
f_{15}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{16}	9.7159e−03(3.2989e−09)	9.7159e−03(5.0556e−10)	9.7159e−03(2.0662e−09)	9.7159e−03(1.8284e−08)	9.7159e−03(1.0150e−08)
f_{17}	−68.7555(1.6865)	−69.4335(2.0382)	−70.2892(1.2832)	−70.3023(1.7741)	7.0276e+01(1.9770)
f_{18}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{19}	0(0)	0(0)	0(0)	6.21014e+01(4.9921e+01)	1.0755e+02(2.8859e+01)
f_{20}	2.3449e−31(4.3822e−31)	6.4327e−31(2.7080e−30)	1.1374e−27(1.9825e−27)	2.0695e−17(6.4694e−17)	4.8075e−09(9.4017e−09)
f_{21}	1.0142e+04(4.0995e+02)	.0260e+04(5.2737e+02)	1.0318e+04(2.8338e+02)	9.9901e+03(5.3427e+02)	1.0066e+04(5.7724e+02)
f_{22}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{23}	2.8900e−30(9.0965e−30)	1.9630e−30(4.2335e−30)	1.6313e−24(5.1564e−24)	2.5152e−16(3.2054e−16)	2.5019e−08(3.8703e−08)
f_{24}	4.8223e−30(7.3259e−30)	2.5348e−28(6.0648e−28)	5.1140e−25(1.5430e−24)	1.8329e−15(4.6833e−15)	4.2984e−08(6.6490e−08)
f_{25}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{26}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{27}	0(0)	0(0)	0(0)	0(0)	0(0)

Table 4
Results obtained by CNPSO with different values of c_1 and c_2 on 30-dimension.

Fun.	$c_1 = 1.3, c_2 = 1.5$	$c_1 = 1.7, c_2 = 1.6$	$c_1 = 1.8, c_2 = 1.5$	$c_1 = 1.9, c_2 = 1.7$	$c_1 = c_2 = 1.2$
f_1	0(0)	0(0)	0(0)	0(0)	0(0)
f_2	−1.0316(1.3180e−08)	−1.0316(9.8465e−13)	−1.0316(1.0870e−8)	−1.0316(9.2518e−07)	−1.0316(1.4839e−14)
f_3	0(0)	0(0)	0(0)	0(0)	0(0)
f_4	0(0)	0(0)	0(0)	0(0)	0(0)
f_5	0(0)	0(0)	0(0)	0(0)	0(0)
f_6	0(0)	0(0)	0(0)	0(0)	0(0)
f_7	0(0)	0(0)	0(0)	0(0)	0(0)
f_8	0(0)	0(0)	0(0)	0(0)	0(0)
f_9	0(0)	0(0)	0(0)	0(0)	0(0)
f_{10}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{11}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{12}	5.9020e−04(2.6334e−04)	1.3223e−03(6.0887e−04)	1.2327e−03(4.8867e−04)	2.5132e−03(1.0070e−03)	1.1579e−04(7.0635e−05)
f_{13}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{14}	−8.8817e−16(0)	−8.8817e−16(0)	−8.8817e−16(0)	−8.8817e−16(0)	−8.8817e−16(0)
f_{15}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{16}	9.7159e−03(4.1819e−10)	9.7159e−03(5.0556e−10)	9.7159e−03(9.8359e−10)	9.7159e−03(7.7384e−09)	9.9997e−03(1.6674e−09)
f_{17}	−68.1019(1.7262)	−70.8621(2.0646)	−69.9132(1.9192)	−70.1161(2.4421)	−6.221e+01(3.2237)
f_{18}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{19}	0(0)	24.5995(41.2540)	8.1052(25.6310)	69.3664(25.6552)	0(0)
f_{20}	1.7396e−29(4.9628e−29)	6.7767e−22(1.4303e−21)	6.0425e−23(1.5544e−22)	5,4855e−18(1.5226e−17)	3.2441e−30(1.0209e−29)
f_{21}	1.0313e+04(4.4304e+02)	9.9891e+03(6.4293e+02)	9.7914e+03(6.1677e+02)	1.0429e+04(3.9789e+02)	1.0213e+03(9.4667e+02)
f_{22}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{23}	8.6425e−30(1.5877e−29)	1.1416e−20(1.1985e−19)	4.1794e−22(1.0038e−21)	5.8957e−17(1.8270e−16)	5.5074e−31(1.6989e−30)
f_{24}	8.6450e−28(1.8291e−27)	1.8800e−20(5.9289e−20)	3.8721e−21(1.1003e−20)	2.4961e−16(6.8937e−16)	5.1378e−30(1.1069e−29)
f_{25}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{26}	0(0)	0(0)	0(0)	0(0)	0(0)
f_{27}	0(0)	0(0)	0(0)	0(0)	0(0)

obtained by setting $c_1 = c_2 = 1.0$, $c_1 = c_2 = 1.3$ and $c_1 = c_2 = 1.2$ does not change much. But the precision obtained by setting $c_1 = c_2 = 1.2$ is slightly higher than this two cases. Thus, $c_1 = c_2 = 1.2$ is used in this paper.

4.1.3. Parameter settings

In experiment 1, each algorithm independently run 30 times, and the maximum number of iterations is used as the termination condition. For all algorithms, the number of population is 40, and the maximum number of iteration (Itmax) is 2500. The other parameters used in 6 algorithms are given in Table 2.

4.1.4. Experimental results and discussion

The comparison results of CNPSO and other five PSOs are given in Tables 5 and 6, and the best results are marked in boldface. For the dimension are 30, unimodal functions is analyzed first, CNPSO outperforms other PSO variants on most of 12 test functions except for f_1, f_2, f_3, f_4 and f_{11}. For the unimodal function f_1, CNPSO shows a similar performance compared with five algorithms except for PSOGSA. For f_2, CNPSO is slightly worse than the comparison algorithms. But CNPSO performs better than PSO, AIWPSO, and PSOGSA on f_3 and f_4. Although H-PSO-SCAC and CNPSO can both find the optimal value 0 on f_5 and f_9, the value of mean and std of H-PSO-SCAC is worse than CNPSO, so CNPSO is more robust

Table 5
Computation results obtained by CNPSO and 5 PSOs on 30 dimension.

Function	Algorithm	Min	Mean	Std	Time
f_1	CNPSO	**0**	**0**	**0**	35.4643
	AIWPSO	**0**	**0**	**0**	9.4037
	PSOGSA	1.4482e−29	2.2577e−25	1.6823e−25	112.5620
	PSO	**0**	**0**	**0**	9.6172
	PSOd	**0**	**0**	**0**	12.3735
	H-PSO-SCAC	**0**	**0**	**0**	0.9486
f_2	CNPSO	−1.0316	−1.0316	1.4839e−14	40.9391
	AIWPSO	−1.0316	−1.0316	6.7752e−16	10.3881
	PSOGSA	−**1.0316**	−**1.0316**	**6.5852e**−**16**	120.4300
	PSO	−1.0316	−1.0316	6.7752e−16	10.3845
	PSOd	−1.0316	−1.0316	1.1150e−12	12.3113
	H-PSO-SCAC	−1.0316	−1.0316	6.5998e−16	0.9815
f_3	CNPSO	**0**	**0**	**0**	37.1906
	AIWPSO	2.9500	1.9208e+01	9.7826	10.0787
	PSOGSA	1.0427e−19	1.3253e−19	2.0815e−20	120.5220
	PSO	2.4564e−03	2.0049e−02	1.6115e−02	10.0827
	PSOd	**0**	**0**	**0**	12.2564
	H-PSO-SCAC	**0**	**0**	**0**	0.9490
f_4	CNPSO	**0**	**0**	**0**	38.3754
	AIWPSO	6.6087	7.7693	2.1061	9.5747
	PSOGSA	1.2790e−09	1.0000	3.1623	177.9100
	PSO	8.2799e−02	4.8875e−01	2.7384e−01	9.4852
	PSOd	**0**	**0**	**0**	12.1065
	H-PSO-SCAC	**0**	**0**	**0**	0.8455
f_5	CNPSO	**0**	**0**	**0**	41.9640
	AIWPSO	6.0893	1.6183e+0	9.9080e+01	11.4548
	PSOGSA	4.9636e−18	5.1667e+03	5.6755e+03	121.1780
	PSO	1.1234e−01	5.8821e−01	5.5068e−01	11.4911
	PSOd	5.6547e+02	1.3005e+03	4.1520e+02	12.0010
	H-PSO-SCAC	0	9.2181e+02	7.7945e+02	0.9014
f_6	CNPSO	**0**	**0**	**0**	42.9587
	AIWPSO	1.8579	4.3639	1.2848	9.9055
	PSOGSA	5.2156	4.1360e+01	3.3815e+01	122.6540
	PSO	1.7367e−01	3.1948e−01	1.4251e−01	10.1217
	PSOd	1.0058	1.0576	1.7435e−02	12.1909
	H-PSO-SCAC	9.4890e−01	1.1115	4.7791e−02	0.8783
f_7	CNPSO	**0**	**0**	**0**	61.6371
	AIWPSO	1.6294e−01	3.6898e−01	1.7885e	9.5321
	PSOGSA	1.6960e−18	2.6565e−18	6.2349e−19	119.8800
	PSO	1.0053e−01	5.3327e−01	2.8845e−01	9.6993
	PSOd	7.2587e+01	8.7660e+01	7.0560	12.0683
	H-PSO-SCAC	7.1334e+01	9.7494e+01	1.1669e+01	0.8541
f_8	CNPSO	**0**	**0**	**0**	40.8290
	AIWPSO	8.0017e−04	1.6100−02	1.5212e−02	20.9547
	PSOGSA	2.3627e−38	4.0389e−38	1.4891e−38	131.0230
	PSO	4.3555e−04	1.5714e−03	3.7964e−03	20.9078
	PSOd	1.8637e+03	1.9963e+03	2.0721e+02	12.4468
	H-PSO-SCAC	2.6051e+03	4.2991e+03	7.9203e+02	1.1277
f_9	CNPSO	**0**	**0**	**0**	50.2364
	AIWPSO	5.2181e−09	6.9914e−06	1.0357e−06	21.7103
	PSOGSA	8.5348e−16	2.6628e−14	3.3552e−14	132.9170
	PSO	1.4636e−13	2.5673e−10	3.7501e−10	21.8545
	PSOd	2.0108	3.5736	1.9718	12.4434
	H-PSO-SCAC	0	4.8335	6.2739	1.0739

Table 5 (*continued*)

Function	Algorithm	Min	Mean	Std	Time
f_{10}	CNPSO	**0**	**0**	**0**	55.5915
	AIWPSO	1.9018e+04	6.8192e+05	7.7438e+05	21.4608
	PSOGSA	9.4243e−09	4.0236e+07	5.9052e+07	132.0740
	PSO	2.4286e+02	1.4403e+03	1.2831e+03	22.2171
	PSOd	1.7872e+04	3.4729e+04	1.0035e+04	12.2981
	H-PSO-SCAC	1.8408e+03	3.9688e+02	1.1727e+02	1.2077
f_{11}	CNPSO	**0**	**0**	**0**	21.9011
	AIWPSO	1	3.9333	1.3113	12.3269
	PSOGSA	0	6.6667e−01	1.2293	118.3250
	PSO	0	0	2.1000	1.5951
	PSOd	1.02e+02	1.0753e+02	1.0605e+01	11.9756
	H-PSO-SCAC	1.0254	2.0211	1.5628	0.8790
f_{12}	CNPSO	**4.7707e−05**	**1.1579e−04**	**7.0635e−05**	53.4128
	AIWPSO	1.2991e−01	3.9982e−01	1.3395e−01	20.6898
	PSOGSA	7.8898e−03	1.8555e−02	9.4974e−03	130.9960
	PSO	5.4254e−02	3.7561e−01	2.6002e−01	20.7414
	PSOd	2.0076e−01	2.4911e−01	7.2978e−01	12.2379
	H-PSO-SCAC	1.7160e−02	4.4752e−02	1.0376e−02	1.1484
f_{13}	CNPSO	**0**	**0**	**0**	39.2005
	AIWPSO	3.5909e+01	7.4243e+01	2.1608e+01	11.9522
	PSOGSA	6.4672e+02	1.2387e+04	3.9158e+05	121.6410
	PSO	1.9990e+01	3.7046e+01	1.3163e+01	11.3463
	PSOd	1.7612e+04	1.8190e+04	6.9897e+02	12.0
	H-PSO-SCAC	1.8411e+03	2.4015e+03	205298e+02	0.8939
f_{14}	CNPSO	**−8.8818e−16**	**−8.8818e−16**	**0**	46.8789
	AIWPSO	3.8861	5.6281	8.6311e−01	12.1095
	PSOGSA	2.2572e−10	8.8945	8.1221	75.7210
	PSO	8.0061e−02	8.9035e−01	9.9656e−01	10.5921
	PSOd	2.1167e+01	2.1386e+01	7.9471e−02	12.0210
	H-PSO-SCAC	1.9999e+01	1.9999e+01	0	0.8973
f_{15}	CNPSO	**0**	**0**	**0**	30.2337
	AIWPSO	9.4588e−01	9.2817e−01	1.2278e−0	11.5943
	PSOGSA	8.7599e−02	7.2619e−01	3.8829e−01	118.8550
	PSO	7.8317–04	3.8656e−02	1.0036e−01	11.5353
	PSOd	9.9964e−01	9.9999e−01	7.8214e−05	12.0317
	H-PSO-SCAC	9.9396e−01	9.9999e−01	1.2194e−03	0.9314
f_{16}	CNPSO	**9.9997e−03**	**9.9997e−03**	**1.6674e−09**	48.8071
	AIWPSO	3.9611e−01	4.2322e−01	1.4956e−02	10.2783
	PSOGSA	3.3121e−01	4.3303e−01	6.1131e−02	119.4640
	PSO	2.2767e−01	3.2644e−01	6.5978e−02	10.2161
	PSOd	2.2789e−01	2.6009e−01	3.1618e−03	11.9308
	H-PSO-SCAC	3.1210	3.5003e−01	1.6172e−02	0.8935
f_{17}	CNPSO	−6.8997e+01	−6.221e+01	3.2237	48.0169
	AIWPSO	−6.6419e+01	−6.3381e+01	1.9451	20.9562
	PSOGSA	**−7.3678e+01**	**−6.6871e+01**	2.3380	131.0660
	PSO	−6.2921e+01	−7.1005e+01	**1.8860e−02**	21.0053
	PSOd	−3.8652e+01	−3.33796e+01	2.9513	12.1495
	H-PSO-SCAC	−6.8674e+01	−6.22655e+01	9.3688	1.244549
f_{18}	CNPSO	**0**	**0**	**0**	46.0322
	AIWPSO	1.4237	4.2226	1.9120	11.2059
	PSOGSA	3.8403	5.1419e+01	309653e+01	110.7540
	PSO	1.4895e−01	1.8744	7.9771e−01	10.9000
	PSOd	2.5761e+01	2.9962e+01	1.9872e+01	11.9307
	H-PSO-SCAC	2.8044e+02	3.3262e+02	2.3658e+01	0.9151
f_{19}	CNPSO	**0**	**0**	**0**	22.8930
	AIWPSO	5.3217e+03	8.5600e+03	2.2659e+06	13.2439
	PSOGSA	1.0900e+02	1.5390e+02	4.4389e+01	122.8490
	PSO	4.3001e+01	7.3111e+01	1.9464e+01	12.7792
	PSOd	1.4503e+03	1.5418e+03	8.5668e+01	12.0466
	H-PSO-SCAC	1.3870e+01	1.6502e+01	8.5668e+01	1.7266
f_{20}	CNPSO	**1.5705e−32**	**3.2441e−30**	**1.0209e−29**	20.1366
	AIWPSO	8.4791e−01	2.3988	1.0667	31.1595
	PSOGSA	1.0952e−21	1.0846	1.1837	141.9840
	PSO	1.0442e−04	1.3737e−03	5.2531e−03	31.0738
	PSOd	1.5209e+01	1.7725e+01	1.8579e+01	12.7048
	H-PSO-SCAC	8.7560e+01	2.4194e+02	7.7709e+01	1.7358
f_{21}	CNPSO	8.8102e+03	1.0213e+03	9.4667e+02	57.5374
	AIWPSO	7.2175e+03	9.5179e+02	7.0209e+02	19.2036
	PSOGSA	**2.9640e+03**	4.9795e+02	7.9770e+02	166

(*continued on next page*)

Table 5 (*continued*)

Function	Algorithm	Min	Mean	Std	Time
	PSO	5.9817e+03	6.0713e+02	**4.9091e+02**	18.0015
	PSOd	7.1898e+03	9.7650e+02	5.2275e+02	15.3367
	H-PSO-SCAC	5.9817e+03	**1.0260e+02**	5.5054e+02	1.8300
f_{22}	CNPSO	**0**	**0**	**0**	45.3870
	AIWPSO	1.1328e−02	2.0604e−03	4.5313e−01	48.4590
	PSOGSA	6.3712e−03	1.6287e−03	5.5303e−02	217.6360
	PSO	1.3948e+02	1.8850e+02	3.4700e+01	53.6819
	PSOd	2.2186e+01	2.7431e+01	1.7512	19.5361
	H-PSO-SCAC	1.3949e−02	3.46608e+01	4.7073e+01	2.2425
f_{23}	CNPSO	**1.3498e−32**	**5.5074e−31**	**1.6989e−30**	21.6703
	AIWPSO	4.2217	1.3370e+01	6.3459	18.9790
	PSOGSA	2.8342e−20	4.1006e+07	1.2512e+08	188.0200
	PSO	1.1886e−02	8.9691e−02	6.9778e−02	19.5928
	PSOd	3.9069e+02	4.9254e+03	2.0183e+03	19.1979
	H-PSO-SCAC	1.1186e−02	1.1087e+08	1.5933e+08	1.9358
f_{24}	CNPSO	**1.4998e−32**	**5.1378e−30**	**1.1069e−29**	42.9633
	AIWPSO	1.0237	2.2220	0.8289	39.6825
	PSOGSA	1.3246e+01	2.6853w + 01	6.9134	208.4720
	PSO	8.2427e−01	1.8099	8.0166e−01	40.9025
	PSOd	1.3754e+02	1.7278e+02	2.3173e+01	19.6120
	H-PSO-SCAC	8.2427e−01	2.0164e+02	3.2306e+01	2.2346
f_{25}	CNPSO	**0**	**0**	**0**	90.1563
	AIWPSO	1.8755e+02	2.8597e+03	1.5141e+03	35.7773
	PSOGSA	1.7113e−17	1.0468e+05	1.2826e+05	213.3360
	PSO	5.9964e−01	3.5821	5.4015	38.6746
	PSOd	9.9197e+05	2.8545e+06	5.7894e+05	22.1320
	H-PSO-SCAC	1.8946e+06	3.1697e+06	5.4814e+05	2.3487
f_{26}	CNPSO	**0**	**0**	**0**	41.9769
	AIWPSO	2.0001e+01	4.9498e+01	1.7816e+01	18.2578
	PSOGSA	1.6293e−05	1.0000e+04	1.0541e+04	190.35
	PSO	5.2364e−02	5.2267e−01	4.4157e−01	21.0633
	PSOd	2.1072e+02	6.9715e+02	8.1090e+02	22.1624
	H-PSO-SCAC	**0**	**0**	**0**	1.7505
f_{27}	CNPSO	**0**	**0**	**0**	43.0440
	AIWPSO	5.2196e−03	8.2742e−03	2.5946e−03	21.3168
	PSOGSA	5.6324e−30	3.4051e−29	3.1151e−29	198.1530
	PSO	5.9121e−03	1.2870e−03	8.5073e−04	17.7363
	PSOd	9.9304e−01	9.9737e−01	3.0166e−03	21.7765
	H-PSO-SCAC	9.9886e−01	9.9990e−01	4.0129e−04	2.3025

Table 6
Computation results obtained by CNPSO and 5 PSOs on 100 dimension.

Function	Algorithm	Min	Mean	Std	
f_1	CNPSO	**0**	**0**	**0**	22.4812
	AIWPSO	**0**	**0**	**0**	11.7301
	PSOGSA	1.7385e−26	205991e−25	2.3509e−25	308.8270
	PSO	**0**	**0**	**0**	11.5733
	PSOd	**0**	**0**	**0**	41.9066
	H-PSO-SCAC	**0**	**0**	**0**	1.4354
f_2	CNPSO	−1.0316	−1.0316	2.3854e−14	29.0481
	AIWPSO	−1.0316	−1.0316	6.7752e−16	18.1668
	PSOGSA	−**1.0316**	−**1.0316**	**6.4539e−16**	336.6330
	PSO	−1.0316	−1.0316	6.7752e−16	41.7699
	PSOd	−1.0316	−1.0316	6.7294e−14	41.7528
	H-PSO-SCAC	−1.0316	−1.0316	3.6029e−16	1.4086
f_3	CNPSO	**0**	**0**	**0**	29.3821
	AIWPSO	3.9011e+02	5.7729e+02	1.3191e+02	16.8545
	PSOGSA	2.1279e−18	1.3000e+04	9.1539e+03	336.6640
	PSO	1.2506	3.4993	2.3536	12.3783
	PSOd	**0**	**0**	**0**	41.8507
	H-PSO-SCAC	**0**	**0**	**0**	1.4347
f_4	CNPSO	**0**	**0**	**0**	31.8084
	AIWPSO	6.3710e+02	7.6983e+02	8.7427e+01	17.4425
	PSOGSA	1.2862e+06	6.4486e+06	3.5321e+05	329.7990
	PSO	7.9716e+02	8.3066e+02	1.9995e+01	11.7792
	PSOd	1.3559e+05	1.1814e+05	9.6934e+04	41.7547
	H-PSO-SCAC	0	1.1681e+02	6.3980e+02	1.3421

Table 6 (*continued*)

Function	Algorithm	Min	Mean	Std	
f_5	CNPSO	**0**	**0**	**0**	47.7468
	AIWPSO	2.1480e+02	4.1648e+03	2.2285e+02	35.4718
	PSOGSA	3.7778e+01	8.6087e+01	4.1554e+01	357.9730
	PSO	3.0491e+01	6.4440e+01	3.3895e+01	30.8636
	PSOd	6.5238e+03	9.0308e+03	2.2555e+03	42.2520
	H-PSO-SCAC	0	9.8954e+03	7.6607e+03	1.8922
f_6	CNPSO	**0**	**0**	**0**	29.9611
	AIWPSO	2.7365e−01	3.4263e−01	3.3601e−02	17.7901
	PSOGSA	1.1734	1.2267	1.4826e−02	342.3730
	PSO	5.3576e−01	4.2091e−01	2.1291e−01	17.0443
	PSOd	1.1729	1.2047	1.33687e−02	42.0284
	H-PSO-SCAC	1.1783	1.2253	1.6288e−02	1.3990
f_7	CNPSO	**0**	**0**	**0**	52.3351
	AIWPSO	5.1212e+01	7.6348e+01	1.3317e+01	17.4328
	PSOGSA	3.0495e−16	7.1685e−16	2.8337e−16	338.7220
	PSO	3.2145e+01	5.2638e+01	1.2582	19.9568
	PSOd	1.1051e+03	1.2979e+03	8.3176e+01	41.8923
	H-PSO-SCAC	1.1163e+03	1.3029e+03	8.4659e+01	1.3742
f_8	CNPSO	**0**	**0**	**0**	56.9842
	AIWPSO	6.2295e+01	1.1403e+01	3.7411e+01	52.6254
	PSOGSA	4.6633e−04	3.6067e+09	3.8878e+09	363.9950
	PSO	2.1502	1.6761e+02	7.9219e+02	47.6792
	PSOd	4.0051e+02	4.2742e+02	5.1758e+01	42.8000
	H-PSO-SCAC	5.5584e+02	6.8465e+02	6.2254e+01	2.3085
f_9	CNPSO	**0**	**0**	**0**	130.5671
	AIWPSO	7.7840e−08	5.7965e−06	1.1766e−05	53.9727
	PSOGSA	1.1093e−12	3.4901e−11	4.9368e−11	378.0040
	PSO	2.6591e−14	7.0320e−09	1.4165e−08	62.6698
	PSOd	2.1549e−03	1.0350e−05	3.3341e−05	42.8165
	H-PSO-SCAC	0	2.6180e+05	7.0483e+05	2.1530
f_{10}	CNPSO	**0**	**0**	**0**	150.3741
	AIWPSO	1.7860e+04	4.1305e+04	1.8222e+04	56.5916
	PSOGSA	2.3737e+01	6.7096e+01	3.3328e+01	376.4120
	PSO	2.9582e+03	1.1261e+04	1.3486e+04	79.1495
	PSOd	1.5249e+05	2.0028e+03	2.3053e+04	42.9203
	H-PSO-SCAC	1.4630e+04	2.2020e+04	3.6765e+03	2.3993
f_{11}	CNPSO	**0**	**2.4667**	**1.5916**	37.6276
	AIWPSO	3.2e+01	5.4933e+01	1.2148e+01	19.6991
	PSOGSA	**2.1e+01**	3.8500e+01	1.0214e+01	342.0920
	PSO	2.3e+01	3.7267e+01	6.7667	13.7221
	PSOd	5.10e+02	5.3287e+02	3.3886e+01	41.9056
	H-PSO-SCAC	6.49e+02	7.2217e+02	3.1562e+01	1.4303
f_{12}	CNPSO	**2.7520e−05**	**1.8163e−04**	**9.8358e−05**	58.1622
	AIWPSO	7.8339e+04	1.3255e+05	4.9557e+04	53.0349
	PSOGSA	1.9571e−01	2.7997e+05	4.3013e+05	372.4820
	PSO	7.0911e+02	4.3831e+01	9.5587e+01	47.4432
	PSOd	4.8728e+01	4.9808e+01	1.2851	42.8140
	H-PSO-SCAC	5.6865e+01	6.9633e+01	6.9072	2.3141
f_{13}	CNPSO	**0**	**0**	**0**	35.487
	AIWPSO	1.0582e+04	1.9183e+04	6.6041e+03	23.3314
	PSOGSA	4.8322e+03	9.3808e+05	4.9682e+05	338.0830
	PSO	7.4436e+02	1.4141e+03	8.5561e+02	18.4942
	PSOd	7.6799e+06	7.8482e+06	3.6226e+05	41.8429
	H-PSO-SCAC	8.8080e+06	9.8192e+06	5.6306e+05	1.5150
f_{14}	CNPSO	−**8.8818e−16**	−**8.8818e−16**	**0**	30.9190
	AIWPSO	1.8897e+01	2.0560e+01	4.5991e−01	24.3837
	PSOGSA	1.9999e+01	1.9999e+01	1.3641e−16	199.7420
	PSO	1.3989e+01	1.6685e+01	1.3190	18.2166
	PSOd	2.1467e+01	2.1528e+01	2.7412e−02	41.8232
	H-PSO-SCAC	1.9999e+01	1.9999e+01	0	1.5277
f_{15}	CNPSO	**0**	**0**	**0**	33.6460
	AIWPSO	2.6867e−01	4.1935e−01	9.3712e−02	23.8794
	PSOGSA	1.3323e−15	7.3052e−01	4.2802e−01	323.8710
	PSO	3.9327e−01	4.3699e−01	3.9996e−02	37.1957
	PSOd	9.9909e−01	9.9993e−01	3.0537e−04	41.8356
	H-PSO-SCAC	1.0000	1.0010	1.4456e−03	1.5699

(*continued on next page*)

Table 6 (*continued*)

Function	Algorithm	Min	Mean	Std	
f_{16}	CNPSO	**9.7151e−03**	**9.7159e−03**	**1.4187e−09**	35.7365
	AIWPSO	1.2699e−01	1.4919e−01	2.5821e−02	18.4788
	PSOGSA	4.2972e−01	4.5239e−01	7.2044e−03	334.2460
	PSO	1.7822e−01	2.1482e−01	4.2529e−02	12.5720
	PSOd	4.4200e−01	4.4633e−01	7.4740e−03	41.6944
	H-PSO-SCAC	4.5979e−01	4.6807e−01	3.6713e−03	1.3932
f_{17}	CNPSO	−7.0607e+01	−6.9299e+01	2.2188	78.2322
	AIWPSO	−6.9712e+01	−6.2861e+01	3.07006	38.0362
	PSOGSA	−**7.2678e+01**	−**6.7337e+01**	2.0686	194.6670
	PSO	−6.7965 + 01	−6.6769e+01	8.2803e−01	28.9032
	PSOd	−2.7502e+01	−2.3388e+01	2.1729	42.6038
	H-PSO-SCAC	−6.7118e+01	−6.5027e+01	**9.9032e−01**	2.3108
f_{18}	CNPSO	**0**	**0**	**0**	40.7877
	AIWPSO	1.0425e+01	2.0764e+01	7.0281	15.2268
	PSOGSA	3.8578	5.8970e+01	3.7391e+01	165.4820
	PSO	2.3759e+01	7.6163e+01	2.9191e+01	15.6897
	PSOd	9.8880e+02	1.1823e+03	5.7196e+01	41.9276
	H-PSO-SCAC	1.1979e+03	1.2752e+03	4.9405e+01	1.5384
f_{19}	CNPSO	**0**	**0**	**0**	36.6369
	AIWPSO	2.7880e+03	6.9040e+03	3.2176e+03	18.3888
	PSOGSA	5.06e+02	7.5985e+04	1.3330e+05	188.3640
	PSO	2.5001e+03	8.6767e+03	1.1891e+02	18.8084
	PSOd	5.4127e+06	5.6191e+06	2.5437e+05	41.8994
	H-PSO-SCAC	5.9408e+06	6.6477e+06	3.6534e+05	1.5788
f_{20}	CNPSO	**5.8383e−21**	**7.5717e−03**	**4.2733e−02**	79.5128
	AIWPSO	2.6412e−01	1.0330	6.5471e−01	65.5393
	PSOGSA	1.3755e−21	1.6801	2.2770	213.1510
	PSO	8.4493e−04	1.0343e−01	2.9607e−01	43.5712
	PSOd	9.8671e+03	1.0537e+03	9.4957	43.8244
	H-PSO-SCAC	9.8630e+02	1.3525e+02	1.6680	73.9729
f_{21}	CNPSO	3.4218e+04	3.7709e+04	9.8367e+02	107.2880
	AIWPSO	**3.4691e+03**	**3.6604e+03**	**7.2160e+02**	52.9178
	PSOGSA	1.6869e+04	2.0301e+04	1.7900e+03	420.7540
	PSO	2.3336e+04	2.6581e+04	6.0134e+03	40.2041
	PSOd	3.5395e+03	3.7812e+03	9.2017e+02	32.7777
	H-PSO-SCAC	2.3335e+04	3.7451e+04	9.8300e+02	3.4740
f_{22}	CNPSO	**0**	**0**	**0**	135.7791
	AIWPSO	9.3399e−02	1.3114e−03	1.4525e−02	134.6886
	PSOGSA	7.7940e−02	1.0576e−03	2.2349e−02	583.2720
	PSO	1.4084e+03	1.6947e+03	1.8761e+02	192.1514
	PSOd	1.0746e+03	1.2411e+03	4.0518e+02	65.2326
	H-PSO-SCAC	1.4084e−03	1.4589e+02	1.1171e+03	5.2094
f_{23}	CNPSO	**1.3498e−32**	**1.3175e−29**	**2.4674e−29**	48.5692
	AIWPSO	8.7505e+01	1.7286e+02	4.1996e+01	39.9563
	PSOGSA	1.3654e+02	3.0071e+03	2.6031e+03	490.5600
	PSO	2.6945	2.8999e+01	5.2963e+01	48.0476
	PSOd	3.4855e+03	3.6703e+03	3.9744e+02	63.6741
	H-PSO-SCAC	2.6945	5.1735e+03	5.4234e+03	4.7731
f_{24}	CNPSO	**1.6251**	**2.7338**	**7.6848e−01**	116.7649
	AIWPSO	1.6177e+01	2.4747e+01	5.9684	95.2068
	PSOGSA	7.7052e+01	1.1915e+02	2.4122e+01	457.6240
	PSO	1.1995e+02	1.4397e+01	1.3062	112.1298
	PSOd	7.4039e+02	7.7707e+02	6.6100e+01	63.0402
	H-PSO-SCAC	1.1995e+01	9.6106e+02	8.8185e+01	16.0628
f_{25}	CNPSO	**0**	**0**	**0**	228.5187
	AIWPSO	2.7377e+04	4.7677e+04	1.2244e+04	80.9809
	PSOGSA	15276e+04	9.7334e+05	9.8320e+05	554.8570
	PSO	8.9038e+01	3.9348e+02	9.3396e+02	78.8694
	PSOd	1.1876e+07	1.3199e+07	3.8166e+05	73.0477
	H-PSO-SCAC	1.3607e+07	1.5798e+07	1.1745e+06	6.8607
f_{26}	CNPSO	**0**	**0**	**0**	104.1390
	AIWPSO	5.4105e+01	9.5306e+01	2.2589e+01	44.2068
	PSOGSA	9.8445e−01	1.0059e+01	1.1781e+01	517.4610
	PSO	3.3162	2.1231e+01	1.9690e+01	62.3106
	PSOd	6.8225e+02	9.2489e+02	4.4075e+02	72.3406
	H-PSO-SCAC	**0**	**0**	**0**	9.2676
f_{27}	CNPSO	**0**	**0**	**0**	120.7780
	AIWPSO	3.2192e−03	4.0575e−03	7.5322e−04	68.4015
	PSOGSA	2.4424e−15	6.3499e−01	4.7718e−01	474.4250
	PSO	1.8133e−02	7.2021e−02	5.5290e−02	34.4007
	PSOd	9.9999e−01	1.0000	1.0170e−04	74.7486
	H-PSO-SCAC	1.0000	1.0000	.8144e−04	3.7884

than other five algorithms. Such advantage of CNPSO can also be proved on f_{11}. It is noticeable that CNPSO can find the global optimal value of 11 out of 12 functions. Though CNPSO doesn't reach the optimal value of f_{12}, the precision is highest among all the comparison algorithm.

When compared on multimodal functions, CNPSO is superior to five comparison algorithms on 12 test functions except for f_{17} and f_{21}. As the test functions becomes more complex, CNPSO can still obtain a high quality solution. For example, Griewank function(f_{15}) has many local optimums, PSO and other PSO vari-

Table 7
The results of comparing CNPSO with other five algorithms on 30 dimension.

Function	PSOvsCNPSO	AIWPSOvsCNPSO	PSOGSAvsCNPSO	PSOdvsCNPSO	H-PSO-SCACvsCNPSO
f_1	=	=	+	=	=
f_2	–	–	–	+	–
f_3	+	+	+	=	=
f_4	+	+	+	=	=
f_5	+	+	+	+	+
f_6	+	+	+	+	+
f_7	+	+	+	+	+
f_8	+	+	+	+	+
f_9	+	+	+	+	+
f_{10}	+	+	+	+	+
f_{11}	+	+	+	+	+
f_{12}	+	+	+	+	+
f_{13}	+	+	+	+	+
f_{14}	+	+	+	+	+
f_{15}	+	+	+	+	+
f_{16}	+	+	+	+	+
f_{17}	+	+	–	+	+
f_{18}	+	+	+	+	+
f_{19}	+	+	+	+	+
f_{20}	+	+	+	+	+
f_{21}	–	–	–	–	–
f_{22}	+	+	+	+	+
f_{23}	+	+	+	+	+
f_{24}	+	+	+	+	+
f_{25}	+	+	+	+	+
f_{26}	+	+	+	+	=
f_{27}	+	+	+	+	+
$b/w/s$	24/2/1	24/2/1	24/3/0	23/1/3	21/2/4

Table 8
The results of comparing CNPSO with other five algorithms on 100 dimension.

Function	PSOvsCNPSO	AIWPSOvsCNPSO	PSOGSAvsCNPSO	PSOdvsCNPSO	H-PSO-SCACvsCNPSO
f_1	=	=	+	=	=
f_2	–	–	–	+	–
f_3	+	+	+	=	=
f_4	+	+	+	+	+
f_5	+	+	+	+	+
f_6	+	+	+	+	+
f_7	+	+	+	+	+
f_8	+	+	+	+	+
f_9	+	+	+	+	+
f_{10}	+	+	+	+	+
f_{11}	+	+	+	+	+
f_{12}	+	+	+	+	+
f_{13}	+	+	+	+	+
f_{14}	+	+	+	+	+
f_{15}	+	+	+	+	+
f_{16}	+	+	+	+	+
f_{17}	+	+	–	+	+
f_{18}	+	+	+	+	+
f_{19}	+	+	+	+	+
f_{20}	+	+	+	+	+
f_{21}	–	–	–	–	–
f_{22}	+	+	+	+	+
f_{23}	+	+	+	+	+
f_{24}	+	+	+	+	+
f_{25}	+	+	+	+	+
f_{26}	+	+	+	+	=
f_{27}	+	+	+	+	+
$b/w/s$	24/2/1	24/2/1	24/3/0	24/1/2	22/2/3

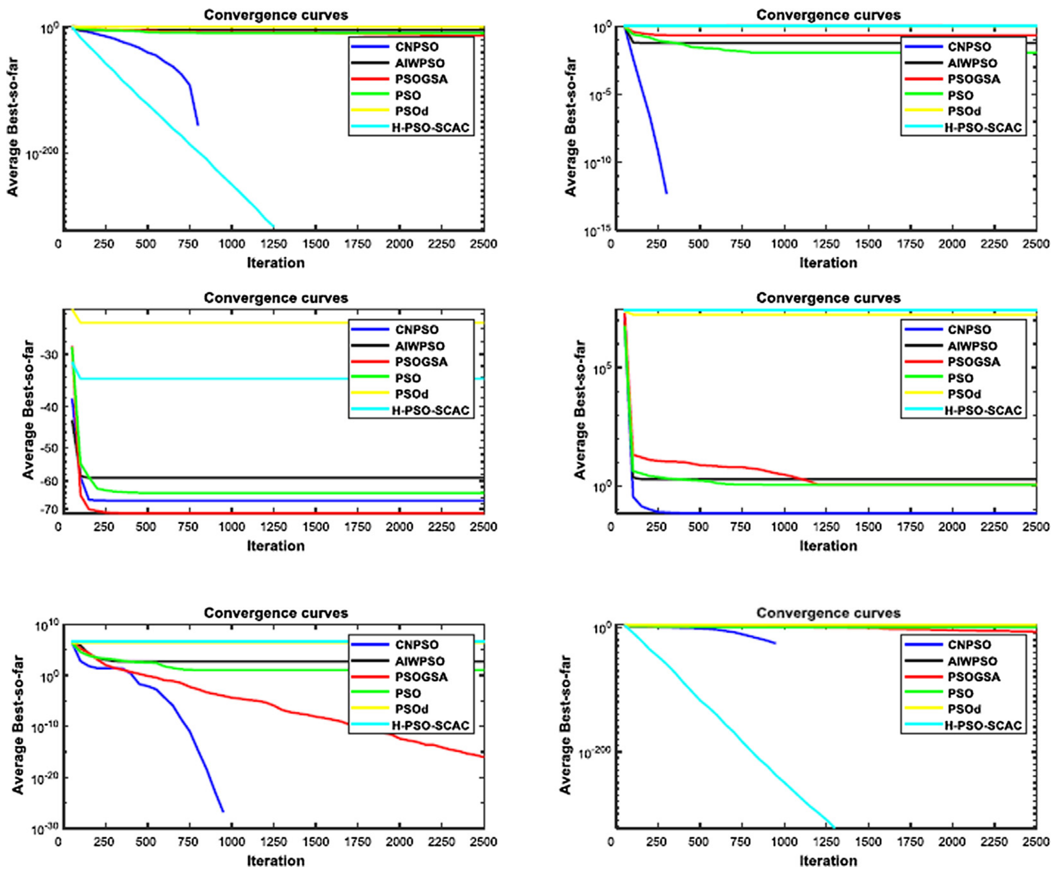

Fig. 2. Convergence rates of 6 functions on 30 dimension.

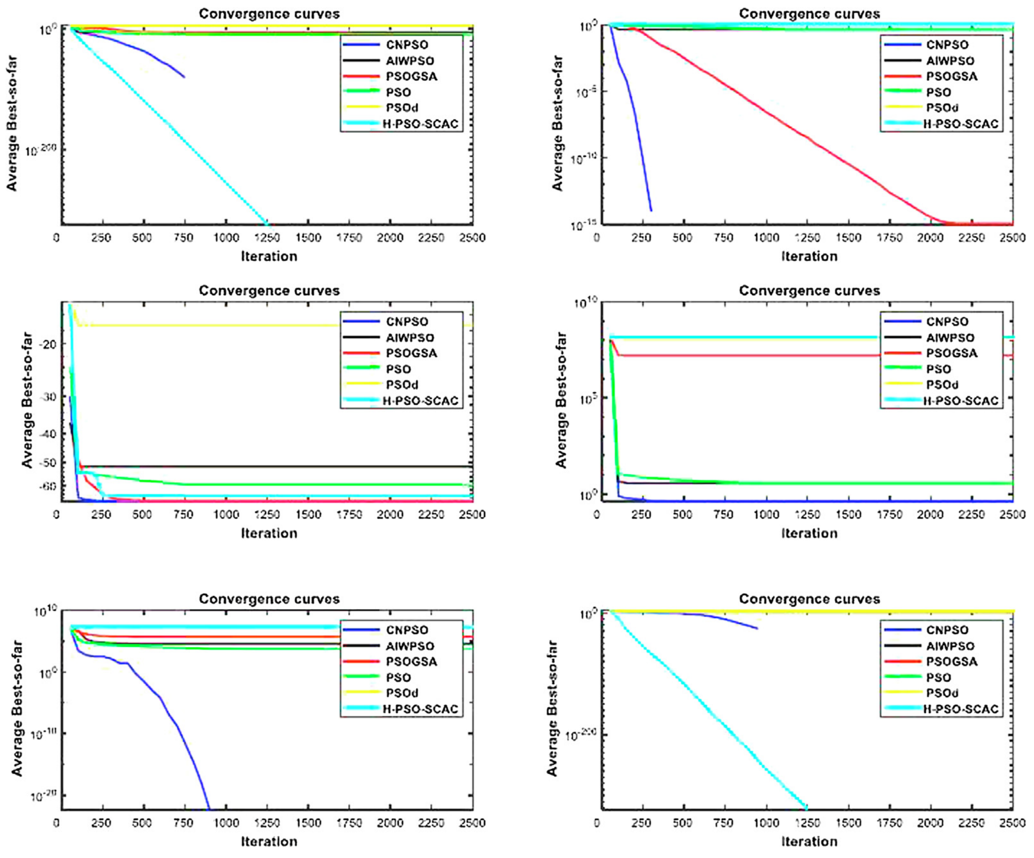

Fig. 3. Convergence rates of 6 functions on 100 dimension.

Table 9
16 benchmark functions in experiment 2.

Functions	Range	Optimal value
$f_1 = \sum_{i=1}^{D} x_i^2$	$[-100, 100]$	0
$f_2 = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}}$	$[-100, 100]$	0
$f_3 = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]$	0
$f_4 = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	$[-1, 1]$	0
$f_5 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]$	0
$f_6 = \max_i\{\lvert x_i \rvert, 1 \leqslant i \leqslant D\}$	$[-100, 100]$	0
$f_7 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-1.28, 1.28]$	0
$f_8 = \sum_{i=1}^{D} i x_i^4$	$[-1.28, 1.28]$	0
$f_9 = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]$	0
$f_{10} = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]$	0
$f_{11} = \begin{cases} \sum_{i=1}^{D} (x_i^2 - 10\cos(2\Pi x_i) + 10), & \lvert x_i \rvert < 0.5 \\ \sum_{i=1}^{D} ((\frac{random(2x_i)}{2})^2 - 10\cos(\Pi random(2x_i)) + 10), & \lvert x_i \rvert \geqslant 0.5 \end{cases}$	$[-5.12, 5.12]$	0
$f_{12} = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-600, 600]$	0
$f_{13} = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{D} x_i^2 / D}) - \exp(\sum_{i=1}^{D} \cos(2\pi x_i / D) + 20 + e$	$[-32, 32]$	0
$f_{14} = \frac{\Pi}{D} 10\sin^2(\Pi y_1) + \frac{\Pi}{D} \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 sin^2(\Pi y_{i+1})]$		
$+ \frac{\Pi}{D} (y_D - 1)^2 + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$		
$y_i = 1 + \frac{x_i + 1}{4}$		
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50, 50]$	0
$f_{15} = \sum_{i=1}^{D} \lvert x_i sin(x_i) + 0.1 x_i \rvert$	$[-10, 10]$	0
$f_{16} = 0.5 + \frac{sin(\sqrt{\sum_{i=1}^{D} x_i^2})^2 - 0.5}{(1 + 0.001 \sum_{i=1}^{D} x_i^2)^2}$	$[-100, 100]$	0

Table 10
Computation results obtained by CNPSO and ABCs on 60 dimension.

Function	Algorithm	Min	Mean	Std
f_1	CNPSO	**0**	**0**	**0**
	ABC-1	6.62e−78	1.35e−76	1.89e−76
	ABC-2	3.72e−101	3.87e−100	7.21e−100
	CosABC	7.50e−118	2.27e−116	2.76e−116
	MABC	9.29e−30	6.03e−29	4.31e−29
f_2	CNPSO	**0**	**0**	**0**
	ABC-1	8.03e−75	9.00e−74	1.23e−73
	ABC-2	4.60e−96	1.62e−94	2.06e−94
	CosABC	8.64e−115	2.47e−113	3.50e−113
	MABC	3.65e−26	3.51e−25	2.72e−25
f_3	CNPSO	**0**	**0**	**0**
	ABC-1	2.65e−78	4.36e−77	6.49e−77
	ABC-2	1.13e−102	6.25e−101	1.21e−100
	CosABC	3.45e−119	1.53e−116	6.30e−116
	MABC	5.61e−65	1.39e−21	8.84e−30
f_4	CNPSO	**0**	**0**	**0**
	ABC-1	1.33e−126	1.46e−112	5.7e−112
	ABC-2	8.06e−153	7.32e−138	2.82e−137
	CosABC	**0**	**0**	**0**
	MABC	5.16e−65	3.00e−62	3.87e−62
f_5	CNPSO	**0**	**0**	**0**
	ABC-1	4.23e−40	1.18e−39	5.35e−40
	ABC-2	1.42e−52	7.13e−52	3.91e−52
	CosABC	3.35e−70	2.79e−69	2.20e−69
	MABC	5.30e−16	6.96e−16	1.20e−16
f_6	CNPSO	**0**	**0**	**0**
	ABC-1	5.50e+00	6.91e+00	6.15e−01
	ABC-2	2.02e+00	2.65e+00	3.16e−01
	CosABC	2.73e−03	7.43e−03	4.07e−03
	MABC	2.92e+01	3.77e+01	3.14e+00
f_7	CNPSO	**0**	**0**	**0**
	ABC-1	**0**	**0**	**0**
	ABC-2	**0**	**0**	**0**
	CosABC	**0**	**0**	**0**
	MABC	**0**	**0**	**0**

(*continued on next page*)

Table 10 (*continued*)

Function	Algorithm	Min	Mean	Std
f_8	CNPSO	**0**	**0**	**0**
	ABC-1	5.38e−158	1.31e−155	3.38e−155
	ABC-2	1.49e−209	1.98e−205	0
	CosABC	4.85e−206	3.63e−203	0
	MABC	4.49e−64	5.00e−62	9.38e−62
f_9	CNPSO	**1.416e−05**	**4.879e−05**	**5.364e−05**
	ABC-1	7.51e−02	1.14e−01	1.69e−02
	ABC-2	5.99e−02	7.82e−02	1.26e−02
	CosABC	1.88e−02	2.44e−02	3.88e−03
	MABC	9.20e−02	1.14e−01	1.16e−02
f_{10}	CNPSO	**0**	**0**	**0**
	ABC-1	**0**	**0**	**0**
	ABC-2	**0**	**0**	**0**
	CosABC	**0**	**0**	**0**
	MABC	**0**	**0**	**0**
f_{11}	CNPSO	**0**	**0**	**0**
	ABC-1	**0**	**0**	**0**
	ABC-2	**0**	**0**	**0**
	CosABC	**0**	**0**	**0**
	MABC	**0**	**0**	**0**
f_{12}	CNPSO	**0**	**0**	**0**
	ABC-1	**0**	1.07e−15	5.84e−15
	ABC-2	**0**	**0**	**0**
	CosABC	**0**	**0**	**0**
	MABC	**0**	**0**	**0**
f_{13}	CNPSO	**−8.88e−16**	**−8.88e−16**	**0**
	ABC-1	7.19e−14	8.27e−14	5.94e−15
	ABC-2	6.84e−14	7.59e−14	5.05e−15
	CosABC	2.22e−14	2.71e−14	3.16e−15
	MABC	1.14e−13	1.37e−13	1.24e−14
f_{14}	CNPSO	**7.85e−33**	**7.85e−33**	**1.44e−48**
	ABC-1	**7.85e−33**	**7.85e−33**	2.78e−48
	ABC-2	**7.85e−33**	**7.85e−33**	2.78e−48
	CosABC	**7.85e−33**	**7.85e−33**	2.78e−48
	MABC	1.49e−31	6.19e−31	3.62e−31
f_{15}	CNPSO	**0**	**0**	**0**
	ABC-1	1.01e−10	8.04e−08	1.55e−07
	ABC-2	3.97e−53	2.78e−17	1.17e−16
	CosABC	1.10e−94	8.88e−17	2.66e−16
	MABC	2.55e−16	8.20e−16	4.69e−16
f_{16}	CNPSO	**9.7e−03**	**9.7e−03**	**6.91e−11**
	ABC-1	4.60e−01	4.76r-01	5.66e−03
	ABC-2	4.42e−01	4.65e−01	1.38e−02
	CosABC	1.78e−01	2.32e−01	1.86e−02
	MABC	4.79e−01	4.84e−01	3.62e−03

The corresponding results in bold are the best results.

ants trapped into the local optimum, while CNPSO can find the global optimal value 0. For multi-modal rotational functions f_{25} and f_{27}, it appears that five PSOs are influenced by the coordinate rotation. However, CNPSO can get the global optimum. Besides, both CNPSO and H-PSO-SCAC can find the global optimum on f_{26}. It can be observed that most of comparison algorithms are trapped by the rotated functions, but CNPSO still show a strong search ability.

Table 11
Experimental result obtained by CNPSO, DEs and CAs on 30-dimension.

Fun.	Max.FEs	CNPSO		JADE		JDE		CA		HS-CA	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
f_3	1.5E+5	**0**	**0**	1.80e−60	8.40e−60	2.50e−28	3.50e−28	9.30e−13	2.70e−12	7.80e−41	3.95e−42
f_4	2.0E+5	**0**	**0**	1.80e−25	8.80e−25	1.50e−23	1.00e−23	1.20e−09	3.10e−09	3.52e−21	9.15e−21
f_6	5.0E+5	**0**	**0**	8.20e−24	4.00e−23	1.40e−15	3.00e−02	3.00e−02	6.45e−53	7.82e−52	
f_8	3.0E+5	**0**	**0**	1.19e−133	5.93e−133	5.30e−104	1.06e−103	2.00e−62	6.34e−62	4.73e−68	9.51e−67
f_{11}	1.5E+5	**0**	**0**	**0**	**0**	**0**	**0**	1.50e−03	1.50e−03	**0**	**0**
f_{12}	3.0E+5	**1.47e−04**	**8.28e−05**	6.40e−04	2.50e−04	3.30e−03	8.50e−04	4.80e−03	1.50e−03	6.92e−02	7.61e−02
f_{13}	5.0E+5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{14}	2.0E+5	**−8.88e−16**	**0**	4.40e−15	**0**	4.70e−15	9.60e−16	1.50e−06	4.60e−06	2.85e−07	8.55e−07
f_{15}	3.0E+5	**0**	**0**	**0**	**0**	**0**	**0**	1.08e−03	7.64e−03	3.10e−08	3.94e−11
f_{18}	5.0E+5	**0**	**0**	5.78e−12	2.47e−12	9.04e−16	6.23e−15	3.42e−12	5.09e−13	3.17e−15	1.66e−15
f_{20}	1.5E+5	**8.42e−33**	**2.17e−48**	1.60e−32	5.50e−48	2.60e−29	7.50e−29	2.27e+01	8.75e+01	3.43e−22	6.89e−24

The corresponding results in bold are the best results.

For the dimension are 100, CNPSO outperforms other PSO variants on most of 12 unimodal functions except for f_1, f_2, f_3. For $f_1 - f_3$, the comparative results is the same as that on 30 dimension. CNPSO and H-PSO-SCAC can both find the optimal value on f_4, f_5 and f_9, but CNPSO is more robust than H-PSO-SCAC when compared with the value of mean and std. Moreover, CNPSO can still find the global optimal value of 11 out of 12 unimodal functions as the dimension increase. When compared on multimodal functions, CNPSO is superior to five comparison algorithms on 12 test functions except for f_{17} and f_{21}. For f_{11}, f_{20}, f_{23} and f_{24}, although the precision of solution found by CNPSO is not higher than that in 30 dimension, CNPSO still perform best in all comparison algorithms. For the multi-modal rotational functions, CNPSO can find the global optimal value 0 compared with other 5 functions on f_{25} and f_{27}. Both CNPSO and H-PSO-SCAC can find the global optimum on f_{26} with 100 dimension.

Based on above analysis, CNPSO outperforms five algorithms on both dimension size. It's worth noticing that CNPSO can find the global optimal solution of 19 functions on 27 test functions. For the multimodal functions, CNPSO show a good performance, while other algorithms fall into the local optimum. The main reason is that the replacement operation of *gbest* can help the algorithm get rid of local optimum efficiently, and the best solution in population is considered differently with other particles, which is not distinguished in five comparison algorithms. Besides, the opposition learning strategy is adopted on the convex combination of best solution and a randomly selected particle in CNPSO. This strategy can guarantee efficient searching of the whole search space and maintain the diversity of algorithm. Therefore, CNPSO has a high search accuracy and show a strong global search ability.

In addition, the computational cost of each comparison algorithms in experiment 1 are given in Tables 5 and 6. From the running time we can see, although CNPSO is not fastest, it performs better than comparison algorithms on most functions. So CNPSO can find a high quality solution in relatively little time.

In order to analyse whether the comparison results got by CNPSO and other PSOs have significant differences, the Wilcoxon signed-rank test at $\alpha = 0.05$ level is used in this subsection. In addition, "+" denotes the CNPSO is significantly better than the comparative algorithm, "=" denotes CNPSO and the comparative algorithm have the same searching ability, and "-" denotes CNPSO is significantly worse than the comparative algorithm. Besides b, w and s presents the number of functions that the solution found by CNPSO is better than, worse than and same as the comparative algorithm respectively. The results at 30 and 100 dimensions are given in Tables 7 and 8.

From the data on 30 dimension we can see, CNPSO is significantly better than PSO, AIWPSO, PSOGSA, PSOd, H-PSO-SCAC on 24, 24, 24, 23, 21 functions respectively, and shows a similar search ability on 1, 1, 0, 3, 4 functions and an inferior search ability on 2, 2,

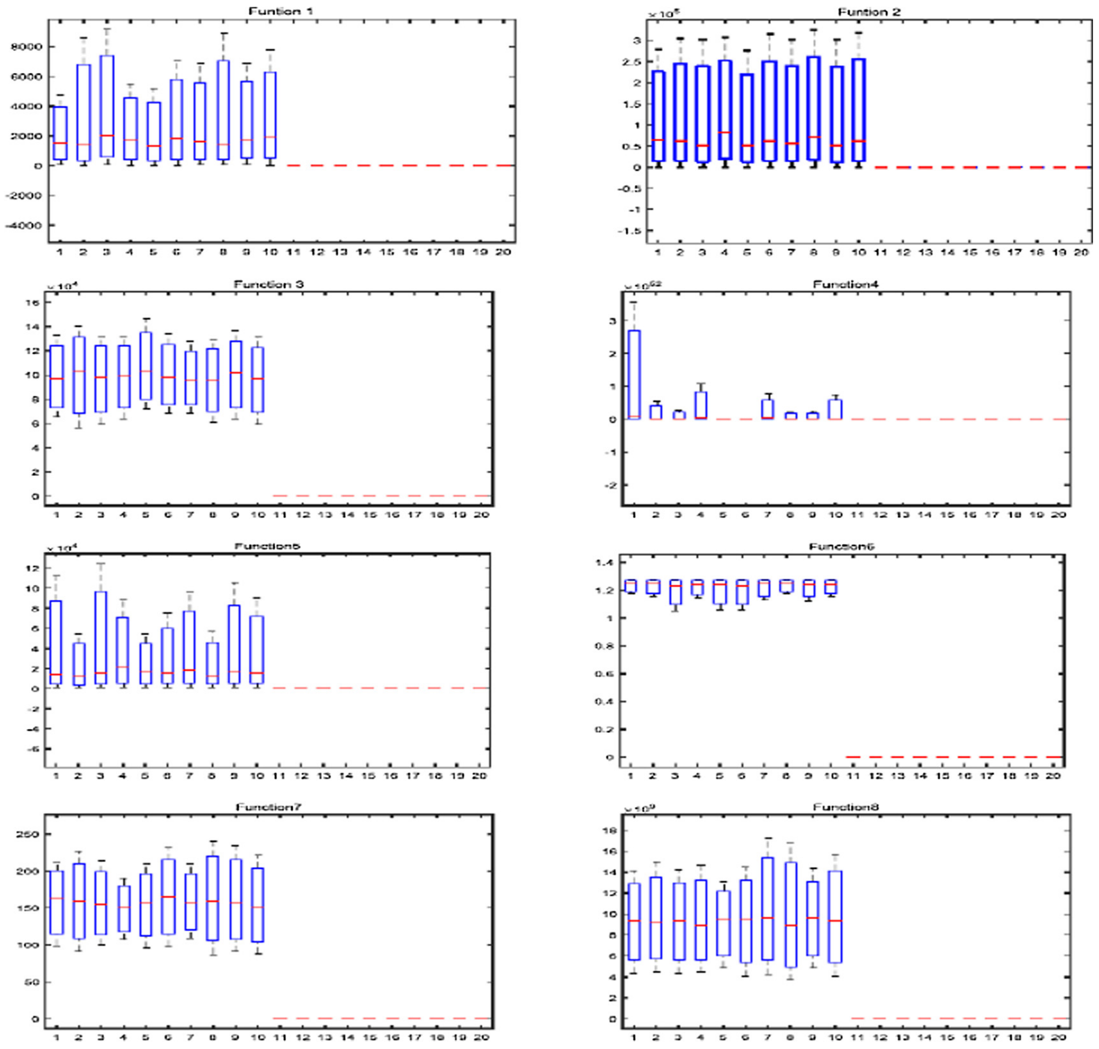

Fig. 4. Box plot of the initial solutions and optimized solution on $f_1 - f_8$.

3, 1, 2 functions respectively. When CNPSO compared with all comparative algorithms, it only lose on 3 functions (f_2, f_{17}, f_{21}). Besides, CNPSO totally win all PSO variants 116 times and are beat 10 times on 27 functions.

From the data on 100 dimension we can see, CNPSO is significantly better than PSO, AIWPSO, PSOGSA, PSOd, H-PSO-SCAC on 24, 24, 24, 24, 22 functions, and shows a similar search ability on 2, 1, 0, 2, 3 functions and an inferior search ability on 2, 2, 3, 1, 2 functions respectively. When CNPSO compared with all comparative algorithms, it only lose on 3 functions (f_2, f_{17}, f_{21}). Besides, CNPSO totally win all PSO variants 118 times and are only beat 10 times on 27 functions.

For 30 and 100 dimension, CNPSO totally beat all PSO variants 234 times on $f_1 - f_{27}$ functions, and performs badly 20 times on all test functions. Based on the result analysis, it indicates CNPSO showed a statistically significant performance compared with all PSO variants on most of the test functions.

4.1.5. Convergent curve of experiment 1

To verify the convergence speed of CNPSO, it is compared with 5 algorithms on 6 functions (f_9, f_{15}, f_{17}, f_{20}, f_{25}, f_{26}). The convergence rate graphs of functions on 30 and 100 dimensions are given in Figs. 2 and 3. From the convergent graph, for f_9, f_{15}, f_{25} and f_{26}, CNPSO can find the optimal value in a faster speed compared with other five PSOs on two dimensions. Though the convergence speed of CNPSO algorithm is not fastest on f_{20}, the precision of obtained solution is higher than five PSOs. For f_{17}, CNPSO performs badly compared with PSOGSA on 30 and 100 dimensions. On the whole, CNPSO can converge to a solution with high quality relatively quickly.

4.2. Experiment 2: Comparison with other ABCs

To further test the performance of CNPSO algorithm, it is compared with some variants of ABC algorithm [41]: ABC-1, ABC-2, CosABC and MABC. 16 benchmark functions with unimodal and multi-modal functions (see Table 9) on 60 dimension are used to test the search ability of each algorithm. For the sake of fairness, the stopping criterion and the number of population are same as those in [41]. The experimental results except for CNPSO are got from [41] directly. The comparative results are given in Table 10.

Based on Table 10, the search capability of CNPSO is stronger than ABCs on 16 functions except for f_4, f_7, f_{10}, f_{11}, f_{12}. For f_4, CNPSO show a similar performance with CosABC, but better than other algorithms. CNPSO and all ABC variants can reach the global optimal value 0 on f_7, f_{10} and f_{11}. And CNPSO is better than ABC-I on f_{12}. From the results we can draw a conclusion that CNPSO is competitive compared with all ABC variants in experiment 2.

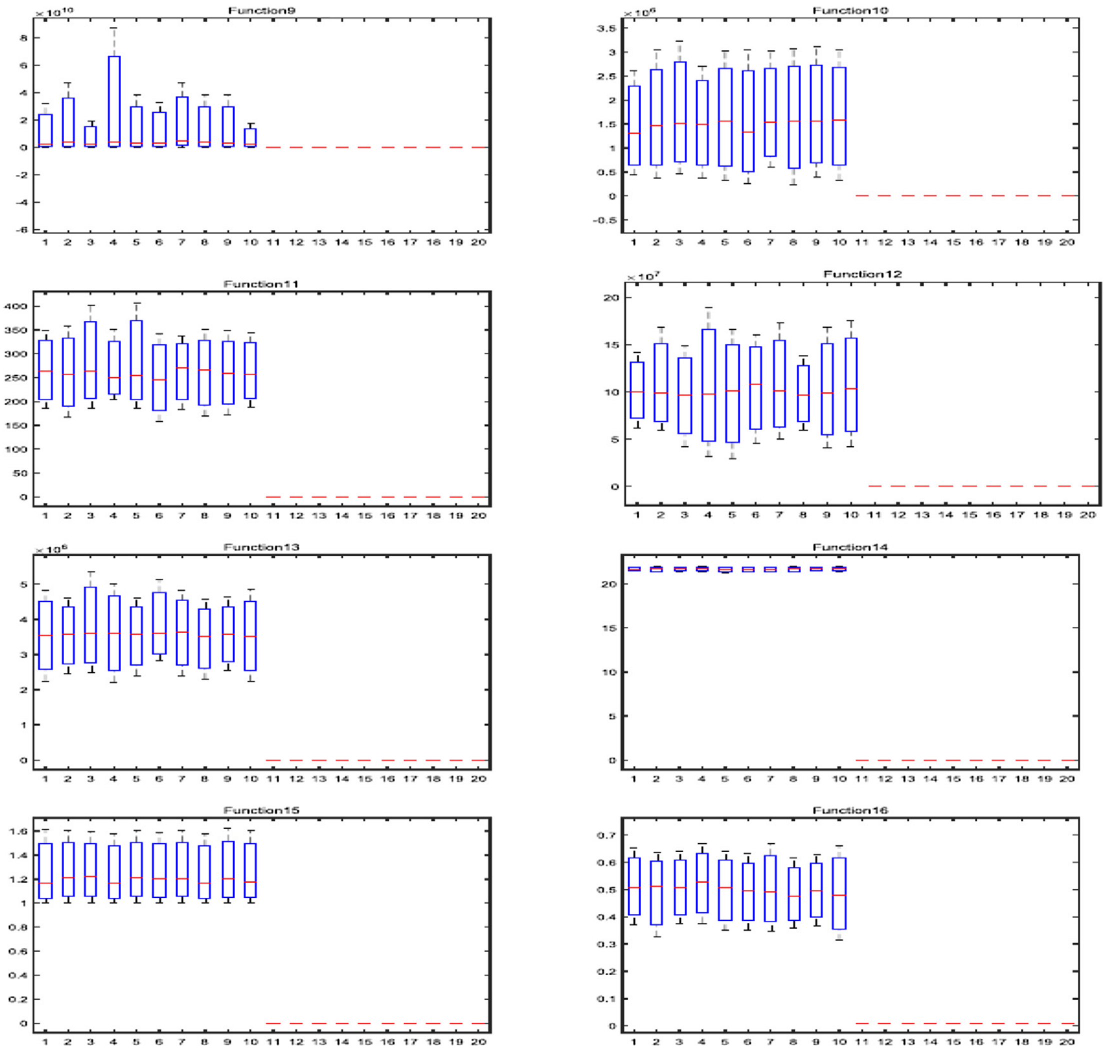

Fig. 5. Box plot of the initial solutions and optimized solution on $f_9 - f_{16}$.

4.3. Experiment 3: Comparison with DEs and CAs

In this subsection, CNPSO and other improved DE and CA algorithms are compared on 11 functions that are selected from experiment 1 (f_3, f_4, f_6, f_8, f_{11}-f_{15}, f_{18}, f_{20}). The dimension of test functions is set 30. Similarly, the experimental results of all algorithms except for CNPSO are got from [42–45] directly and given in Table 11.

Based on the data in Table 11. CNPSO outperforms DEs and CAs on all functions except for f_{11}, f_{13} and f_{15}. For f_{11}, CNPSO performs as well as JADE, JDE, and HS-CA, but better than CA. And all algorithms can find the global optimal value on f_{13}. Besides, CNPSO is superior to CA and CA-HS on f_{15}. Thus, CNPSO can efficiently find a promising solution.From the result analysis of experiment 2 and experiment 3, CNPSO is better than or show a similar performance compared with ABCs, DEs and CAs, i.e., CNPSO is not beaten by the comparison algorithms on any test functions. It indicated two update operations in proposed algorithm indeed help the algorithm avoid getting stuck or falling into stagnation and enhance the global search ability.

In order to verify the impact of each modification on the exploratory of exploitative behaviors of the algorithm, the Box plot is used in this part to get 27 figures of 20 solutions during optimization before and after applying the new operator. In this part, the proposed algorithm is run 10 times to get 10 initial solutions and 10 optimized solutions. And the first ten boxes represent the initial population, the last ten boxes represent optimized populations in each figures.

Based on the situation of two set of solutions dispersion, we can see that the set of initial solutions has been dispersed by using the update operations except for f_{21}. So the modifications of the proposed algorithm can help algorithm escape local optimum and improve the global search ability (see Figs. 4–7).

5. Computational complexity analysis

In this section, the computational complexity of the proposed algorithm and five PSOs in experiment 1 are analyzed to evaluate the running efficiency of algorithm, and corresponding result of each algorithm are given in Table 12.

For the proposed algorithm, it has only one loops when update the velocity and position of particle, and the main computational cost is to calculate the function value, so the computational complexity of CNPSO is $O(D * N_p)$, which is varies according to the number of population and the dimension of function. From Table 12 we can know that the complexity of CNPSO is the same as all comparison algorithm except for PSOGSA. So CNPSO can improve the performance of algorithm without increasing the extra computational complexity.

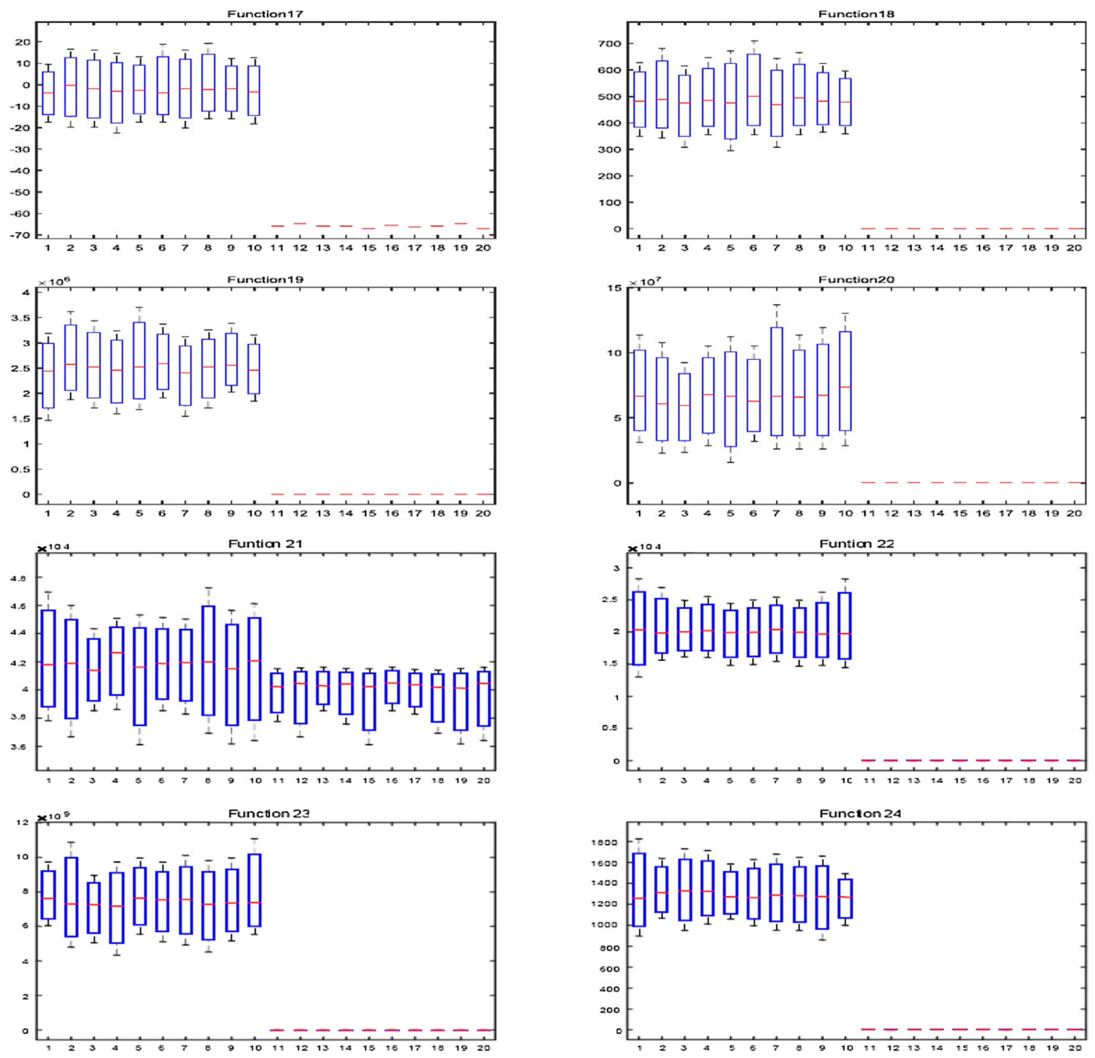

Fig. 6. Box plot of the initial solutions and optimized solution on $f_{17} - f_{24}$.

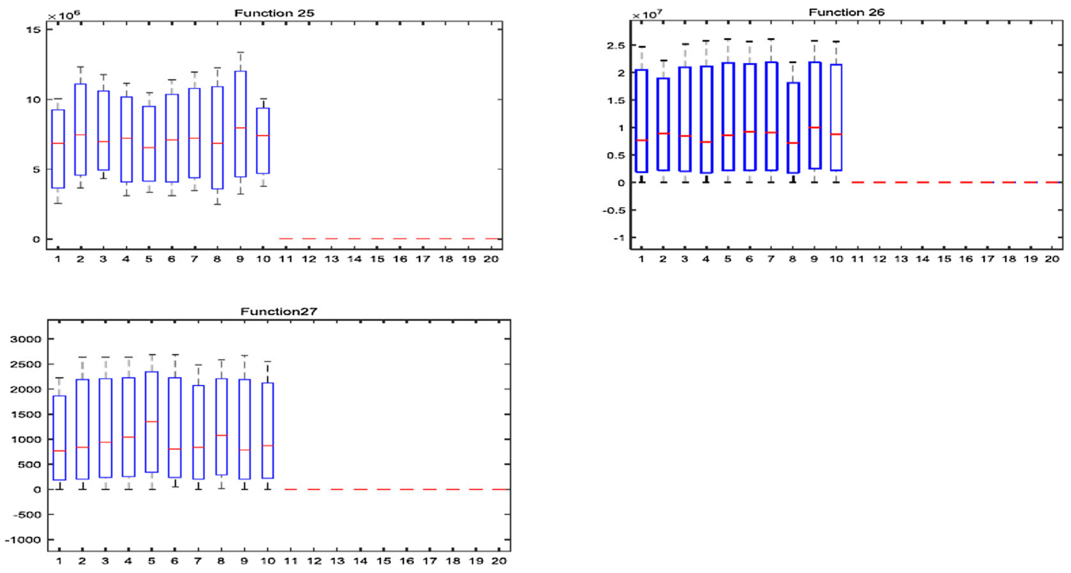

Fig. 7. Box plot of the initial solutions and optimized solution on $f_{25} - f_{27}$.

Table 12
The computational complexity of CNPSO and PSOs.

Algorithm	Initialize	Function calculation	Update process	Total computational complexity
PSO	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$
AIWPSO	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$
PSOGSA	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p^2)$	$O(D * N_p^2)$
CNPSO	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$
PSOd	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$
H-PSO-SCAC	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$	$O(D * N_p)$

Table 13
Optimal results obtained by CNPSO, CPSO, MPM, IOD, SPA and TGA algorithm.

Optimum variables	CPSO	MPM	IOD	SPA	TGA	CNPSO
x_1	0.051728	0.050000	0.053396	0.051480	0.051989	0.050000
x_2	0.357644	0.315900	0.399180	0.351661	0.363965	0.607914
x_3	11.244543	14.250000	9.185400	11.632201	10.890522	2.000000
$f(x)$	0.0126747	0.0128334	0.0127303	0.0127048	0.0126810	**0.006076**

The corresponding results in bold are the best results.

5.1. Structure design of tension-pressure spring

In this subsection, CNPSO algorithm is applied on a practical problem: structure design of tension-pressure spring problems, which minimize the weight of a tension spring under the constraint requirements such as shear stress, frequency of vibration and so on. This problem includes three design variables: coil diameter: $d(x_1)$, average diameter of spring coil: $D(x_2)$ and the effective number of circles: $N(x_3)$.The optimization problem is describe as

$$\min f(x) = (x_3 + 2)x_2x_1^2$$

subject to

$$g_1(x) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0,$$
$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0,$$
$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0,$$
$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0,$$

where

$$0.05 \leqslant x_1 \leq 2,$$
$$0.25 \leqslant x_2 \leq 1.3,$$
$$2 \leqslant x_3 \leq 15,$$

To verify the search ability of the proposed algorithm, CNPSO and some algorithms (CPSO [46], MPM [47], IOD [48], SPA [49], TGA [50]) are compared on this optimization problem. The optimal variable and optimal value obtained by each algorithm are given in Table 13. Besides, the min, mean and std of obtained solution is used as the evaluation index (see Table 14). From the results,

Table 14
Computation results obtained by CNPSO, CPSO, MPM, IOD, SPA and TGA algorithm.

Algorithm	Min	Mean	Std
CPSO	0.0126747	0.012924	5.198500e–005
MPM	0.0128334	NA	NA
IOD	0.0127303	NA	NA
SPA	0.0127048	0.012822	3.939000e–005
TGA	0.0126810	0.012973	5.900000e–005
CNPSO	**0.0060761**	**0.0060761**	**2.879554e–009**

The corresponding results in bold are the best results.

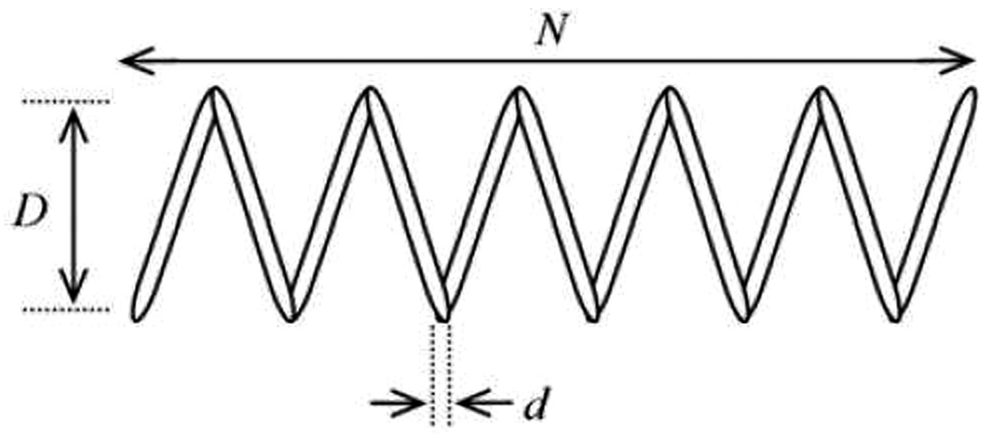

Fig. 8. Structure design of tension-pressure spring.

CNPSO performs best in all comparative algorithms, so we can draw a conclusion it can solve this optimization problem more effectively (see Fig. 8).

6. Conclusion

To improve the performance of basic PSO, an modified PSO algorithm based on velocity updating mechanism is introduced in this paper. Two improvement strategies is proposed to generate a new position for each particle. For the individual who is not best in the population, based on the probability selection strategy, a particle was selected randomly from a set with high quality to replace *gbest* in the velocity formula. For the best individual, the convex combination of best particle and a randomly selected particle is generated, then the opposition learning mechanism were adopted on it to generate a new solution again.

In the improved algorithm, the *gbest* is replaced by a high quality particle that is randomly selected by the probability calculation. The randomness of select can let particle make full use of good information of other high quality of particles. So this strategy can avoid falling into a local optimum or happening the stagnation phenomenon. Besides, the best particle is also considered in this paper, which is not distinguished in basic PSO. In this update process, the opposition learning mechanism are used on the convex combination of best individual and another particle to generate a new solution. By using this strategy, the algorithm can search more widely in space and find a potential solution, which can enhance the diversity of population and improve the global search ability. Finally, three comparative experiments and a practical problem were carried out to verify the performance of CNPSO, and the results indicate CNPSO performs better compared with other algorithms. Therefore, CNPSO has a good search capability and search efficiency. Besides, the proposed algorithm is compared with some other algorithms in three experiments and a optimization problem to verify the performance. The results indicates the proposed algorithm can jump out the local optimum effectively and improve the global search ability.

In the future, the methods proposed in this paper can be used to other meta-heuristic algorithms. And the improved algorithm can be discussed to handle the problems with discrete variables and multi-objective optimization problems, and to solve the many other practical optimization problems.

References

[1] Aliadeh S, Ghazanfari M. Learning FCM by chaotic simulated anealing. Chaos Solit Fract 2009;41:1182–90.
[2] Yang XS. Firefly algorithm. Nature-Inspired Metah Algor 2008;20:79–80.
[3] Kennedy J, Eberhart R. Particle swarm optimization. Proc IEEE Int Conf Networks 1995;4:1942–8.
[4] Yang XS, Deb S. Cuckoo search via levy flights. In: Proceedings of world congress on nature and biologically inspired computing. p. 210–4.
[5] Guan X, Xin YY. An improved artificial fish swarm algorithm and its application. International conference on electrical engineering and automatic control, vol. 10. p. 57–9.
[6] Zhang RY, Yun WY, Kopfer H. Heuristic-based truck scheduling for inland container transportation. Or Spectrum 2010;32:787–808.
[7] Zeng R, Wang YY. A chaotic simulated annealing and particle swarm improved artificial immune algorithm for flexible job shop scheduling problem. Eurasip J Wirel Commun Network 2018;101:1–10.
[8] Valdez F, Vazquez JC, Melin P, Castillo O. Comparative study of the use of fuzzy logic in improving particle swarm optimization variants for mathematical functions using co-evolution. Appl Soft Comput 2017;52:1070–83.
[9] Olivas F, Valdez F. Dynamic parameter adaptation in particle swarm optimization using interval type-2 fuzzy logic. Soft Comput 2016;20 (3):1057–70.
[10] Melin P, Olivas F, et al. Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. Exp Syst Appl 2013;40(8):3196–206.
[11] He M, Liu M, et al. Particle swarm optimization with damping factor and cooperative mechanism. Appl Soft Comput 2019;76:45–52.
[12] Chen Y, Li L, et al. Particle swarm optimizer with crossover operation. Eng Appl Artif Intell 2018;70:159–69.
[13] Tsai HC. Unified particle swarm delivers high efficiency to particle swarm optimization. Appl Soft Comput 2017;55:371–83.
[14] Pan A, Wang L, et al. A diversity enhanced multiobjective particle swarm optimization. Inform Sci 2018:441–65.
[15] Ye WX, Feng WJ, Fan SH. A novel multi-swarm particle swarm optimization with dynamic learning strategy. Appl Soft Comput 2017;61:832–43.
[16] Yang CM, Simon D. A new particle swarm optimization technique. Int Conf Syst Eng 2005;5:164–9.
[17] Wang L, Yang B, Orchard J. Particle swarm optimization using dynamic tournament topology. Appl Soft Comput 2016;48:584–96.
[18] Shi Y, Liu HC, et al. Cellular particle swarm optimization. Inform Sci 2011;181:4460–93.
[19] Al-bahrani LT, Patra JC. A novel orthogonal PSO algorithm based on orthogonal diagonalization. Swarm Evol Comput 2017;15:1–23.
[20] Yan BL, Zhou YC, et al. A particle swarm optimization algorithm with random learning mechanism and Levy flight for optimization of atomis clusters. Comput Phys Commun 2017;219:79–86.
[21] Fang W, Sun J, et al. A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population. Inform Sci 2016;330:19–48.
[22] Gou J, Guo WP, et al. A novel improved particle swarm optimization algorithm based on individual difference evolution. Appl Soft Comput 2017;57:468–81.
[23] Zhang K, Huang QJ, Zhang YM. Enhancing comprehensive learning particle swarm optimization with local optima topology. Inform Sci 2019;471:1–18.
[24] Xu G. An adaptive parameter tuning of particle swarm optimization algorithm. Appl Math Comput 2013;219:4560–9.
[25] Liu ZG, Ji XH, Liu YX. Hybrid non-parametric particle swarm optimization and its stability analysis. Exp Syst Appl 2018;92:256–75.
[26] Chen K, Zhou FY, Liu A. Chaotic dynamic weight particle swarm optimization for numerical function optimization. Knowl-Based Syst 2018;139:23–40.
[27] Zhang LM, Tang YG, et al. A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. Appl Soft Comput 2015;28:138–49.
[28] Li SF, Cheng CY. Particle swarm optimization with fitness adjustment parameters. Comput Ind Eng 2017;113:831–41.

[29] Liang HT, Kang FH. Adaptive mutation particle swarm algorithm with dynamic nonlinear changed inertia weight. Optik-Int J Light Electron Opt 2016;127:8036–42.
[30] Afia AE, Sarhani H, Aoun O. Hidden markov model control of inertia weight adaptation for particle swarm optimization. IFAC PapersOnLine 2017;50:9997–10002.
[31] Yang CW, Gao W, et al. Low-discrepancy sequence initialized particle swarm optimization algorithm with high-order nonlinear time-varying inertia weight. Appl Soft Comput 2015;29:386–94.
[32] Liu H, Cai ZX, Wang Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 2010;10:629–40.
[33] Ali AF, Tawhid MA. A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems. Ain Shams Eng J 2017;8:197–206.
[34] Javidrad F, Nazari M. A new hybrid particle swarm and simulated annealing stochastic optimization method. Appl Soft Comput 2017;60:634–54.
[35] Xia XW, Gui L, et al. A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. J Comput Sci 2017;724:1–13.
[36] Yadav A, Deep K, et al. Gravitational swarm optimizer for global optimization. Swarm Evol Comput 2016;31:64–89.
[37] Nickabadi A, Ebadzadeh MM, Safabakhsh R. A novel particle swarm optimization algorithm with adaptive inertia weight. Appl Soft Comput 2011;11:3658–70.
[38] Mirjalili S, Hashim SZM. A new hybrid PSOGSA algorithm for function optimization. In: International conference on computer and information application. p. 374–7.
[39] Chen K, Zhou FY, et al. A hybrid particle swarm optimizer with sine cosine acceleration coefficients. Inform Sci 2018;422:218–41.
[40] Kiran MS. Particle swarm optimization with a new update mechanism. Appl Soft Comput 2017;60:670–8.
[41] Xiang WL, Li YZ, et al. A novel artificial bee colony algorithm based on the cosine similarity. Comput Ind Eng 2018;115:54–68.
[42] Zhang J, Sanderson AC. JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 2009;13:945–58.
[43] Brest J, Greiner S, et al. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 2006;10:646–57.
[44] Peng B, Reynolds RG. Cultural algorithms: knowledge learning in dynamic environments. Cong Evol Comput 2004;2:1751–8.
[45] Gao XZ, Wang X, et al. A hybrid optimization method for wind generator design. Int J Innov Comput 2012;8:4347–73.
[46] He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 2007;20:89–99.
[47] Belegundu AD. A study of mathematical programming methods for structural optimization. Iowa City, Iowa: Department of Civil and Environmental Engineering, University of Iowa; 1982.
[48] Arora JS. Introduction to optimum design. New York: McGraw-Hill; 1989.
[49] Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113–27.
[50] Coello CAC, Montes EM. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 2002;16:193–203.