Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 743671, 10 pages
http://dx.doi.org/10.1155/2015/743671

Research Article

Parallel and Cooperative Particle Swarm Optimizer for Multimodal Problems

Geng Zhang[1] and Yangmin Li[1,2]

[1]*Department of Electromechanical Engineering, University of Macau, E11-4067, Avenida da Universidade, Taipa, Macau*
[2]*Tianjin Key Laboratory for Advanced Mechatronic System Design and Intelligent Control, Tianjin University of Technology, Tianjin 300384, China*

Correspondence should be addressed to Yangmin Li; ymli@umac.mo

Received 25 November 2014; Accepted 13 February 2015

Academic Editor: Erik Cuevas

Although the original particle swarm optimizer (PSO) method and its related variant methods show some effectiveness for solving optimization problems, it may easily get trapped into local optimum especially when solving complex multimodal problems. Aiming to solve this issue, this paper puts forward a novel method called parallel and cooperative particle swarm optimizer (PCPSO). In case that the interacting of the elements in D-dimensional function vector $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$ is independent, cooperative particle swarm optimizer (CPSO) is used. Based on this, the PCPSO is presented to solve real problems. Since the dimension cannot be split into several lower dimensional search spaces in real problems because of the interacting of the elements, PCPSO exploits the cooperation of two parallel CPSO algorithms by orthogonal experimental design (OED) learning. Firstly, the CPSO algorithm is used to generate two locally optimal vectors separately; then the OED is used to learn the merits of these two vectors and creates a better combination of them to generate further search. Experimental studies on a set of test functions show that PCPSO exhibits better robustness and converges much closer to the global optimum than several other peer algorithms.

1. Introduction

Inspired from social behavior and cognitive behavior, Kennedy and Eberhart [1, 2] proposed particle swarm optimizer (PSO) algorithm to search for optimal value through population-based iterative learning algorithm. Due to the simple implementation and effective searching ability of the PSO algorithm, it has been widely used in feature selection [3], robot path planning [4], data processing [5], and other problems. However, many experiments have shown that the PSO algorithm may easily get trapped into local optimum especially when facing complex multimodal optimization problems. Better optimization algorithms are always needed for solving complex real-world engineering problems. In general, the unconstrained optimization problems that we are going to solve can be formulated as a D-dimensional minimization problem as follows:

$$\text{Min } f(X), \quad X = [x_1, x_2, \ldots, x_d, \ldots, x_D], \tag{1}$$

where $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$ is the vector to be optimized and D is the number of parameters [6].

In PSO, a member in the swarm, called a particle, represents a potential solution which is a point in the search space. Let M denote the size of the swarm, the current state of each particle i is represented by its position vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{id}, \ldots, x_{iD}]$, and the movement of particle i is represented by velocity vector $V_i = [v_{i1}, v_{i2}, \ldots, v_{id}, \ldots, v_{iD}]$, where $i = 1, 2, \ldots, M$ is positive integer indexing particle in the swarm. Using $t = 1, 2, \ldots, T$ represents the iteration number; the velocity $v_{id}(t)$ and the position $x_{id}(t)$ can be updated as follows:

$$\begin{aligned} v_{id}(t+1) &= \omega v_{id}(t) + c_1 r1_{id}\left[p_{id}(t) - x_{id}(t)\right] \\ &\quad + c_2 r2_{id}\left[p_{nd}(t) - x_{id}(t)\right], \\ x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1), \end{aligned} \tag{2}$$

where the inertia weight $\omega \in (0, 1)$ determines how much the previous velocity can be preserved. A large inertia weight value tends to global exploration and a small value for local exploitation. c_1 and c_2 denote the acceleration constants which are usually set 2.0 or adaptively

controlled [7]. $r1_{id}$ and $r2_{id}$ are random numbers generated between 0 and 1 for the dth dimension of ith particle. $P_i(t) = [p_{i1}(t), p_{i2}(t), \ldots, p_{id}(t), \ldots, p_{iD}(t)]$ represents the best previous position of particle i which is defined by *pbest* from the previous t iterations and $P_n(t) = [p_{n1}(t), p_{n2}(t), \ldots, p_{nd}(t), \ldots, p_{nD}(t)]$ represents the best position among particle i's neighborhood which is defined by *gbest* or *lbest* from the previous t iterations. The *gbest* model, which is inclined to exploitation, has a faster convergence speed but has a higher probability of getting stuck into local optimum than the *lbest* model. On the contrary, the *lbest* model considered focusing more on exploration is less vulnerable to the attraction of local optima but has slower convergence speed than the *gbest* model [8]. The position vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{id}, \ldots, x_{iD}]$ and the velocity vector $V_i = [v_{i1}, v_{i2}, \ldots, v_{id}, \ldots, v_{iD}]$ are initialized randomly and are updated by (2) generation by generation until some criteria are met.

There are many modified versions of PSO that have been reported in the literature. Most studies address the performance improvement of PSO from one of the four aspects: population topology [9–11], diversity maintenance [6, 12, 13], hybrid with auxiliary operations [14–16], and adaptive PSO [8, 17]. However, it is difficult to find a robust solution due to the complexity and large search space of high dimensional problems. Therefore, people try to split the large search space into smaller ones in order to simplify the problem. In [18], the search space implemented by genetic algorithm is divided by splitting the solution vector into several smaller vectors. Each smaller search space is optimized separately and the fitness function is evaluated by combining all the solutions found by the smaller search spaces. The same technique is used in PSO called cooperative PSO (CPSO_H), which uses several subswarms to search D-dimensional function vector $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$ separately [19]. The searching results are integrated by a global swarm to improve the performance of the original PSO on high dimensional problems. Compared with traditional PSO, the CPSO_H has significant performance in terms of solution quality and robustness and performs better and better as the dimension of the problem increases. However, the efficiency of the algorithm is highly affected by the degree of interacting between function vectors $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$.

Inspired from previous studies, a novel algorithm called parallel and cooperative PSO (PCPSO) is proposed in this paper. PCPSO tries to overcome the influence of interacting between vector elements when taking similar split method as mentioned in [19]. Assuming that there is no interacting among the elements of the vector $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$, the CPSO is used [19]. Although the application of CPSO is useless in solving real problems, it provides a framework for PCPSO. For high degree of interacting of vector elements, local optimum can be obtained by CPSO [20]. In order to jump out of local optimum, orthogonal experimental design (OED) is used to learn from two locally optimal vectors which are achieved by CPSO [21]. A better combination of these two vectors can be obtained to push the search further to get closer to global optimum.

The rest of this paper is organized as follows. Section 2 describes the PCPSO algorithm. In Section 3, the benchmark test functions are used to test PCPSO algorithm and their results are compared with some peer algorithms taken from literature to verify the effectiveness and robustness of PCPSO algorithm. Final conclusions are given in Section 4.

2. Parallel and Cooperative Particle Swarm Optimizer

2.1. Cooperative Particle Swarm Optimizer. Among PSO variants in the literature, premature convergence is still the main deficiency of the PSO especially when facing large dimensional high correlation problems. The difficulty for particles jump out of local optimum is that particles need to change several elements of vector $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$ simultaneously, but usually the PSO variants can only change one element at one time when facing local optimum. Although the OED method is used to combine the best elements of vector $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$ as an exemplar [21]; the function value $f(X)$ usually gets bigger because of the high correlation. Here, we first introduce the CPSO used in [19]; then the PCPSO is proposed based on CPSO. OED is used to learn and create a new vector from two locally optimal vectors which are achieved by CPSO and the new vector is treated as a start point for further search.

In [19], there are two cooperative PSO algorithms, CPSO_S_k and CPSO_H_k. In CPSO_S_k, the D-dimensional search space is decomposed into k subcomponents, each corresponding to a swarm of s-dimensions (where $n = k \times s$). CPSO_H_k is a hybrid method combining a standard PSO with the CPSO_S_k. The k subcomponents are used to place interacting elements together. However, the interacting elements are not known in real problems. We simplify the CPSO_S_k algorithm as CPSO which means the D-dimensional search space is decomposed into D subcomponents. Algorithm 1 shows the algorithm of CPSO. In order to evaluate the fitness of a particle in a swarm, the context vector is applied, which is a concatenation of all global best particles from all D swarms. The evaluation of the ith particle in the dth swarm is done by calling the function $\mathbf{b}(d, S_d \cdot x_i)$ which returns a D-dimensional vector with its dth component replaced by $S_d \cdot x_i$ [22].

2.2. Parallel and Cooperative Particle Swarm Optimizer Algorithm. Since CPSO method uses one-dimensional swarm to search each dimension and interacting elements of the solution vector cannot be changed simultaneously, it is easier to fall into local optimum. If we try to use the CPSO algorithm (called First1D) to solve the high correlation problem, the vector $X_{L1} = [x_{L11}, x_{L12}, \ldots, x_{L1d}, \ldots, x_{L1D}]$ started from random values will fall into a locally optimal vector. However, when another CPSO algorithm (called Second1D) which does not have any relationship with First1D is also used to solve the same problem, another locally optimal vector $X_{L2} = [x_{L21}, x_{L22}, \ldots, x_{L2d}, \ldots, x_{L2D}]$ will be obtained. Since the performance of CPSO depends on numerical distribution when $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]$ is initialized by random

(1) **Define** $b(d, z) \equiv (S_1 \cdot p_n, S_2 \cdot p_n, \ldots, S_{d-1} \cdot p_n, z, S_{d+1} p_n, \ldots, S_D \cdot p_n)$
(2) Create and initialize D one-dimensional PSOs: S_d, $d \in [1, \ldots, D]$
(3) **Repeat:**
(4) **for** each swarm $d \in [1, \ldots, D]$:
(5) **for** each particle $i \in [1, \ldots, M]$:
(6) **if** $f(\mathbf{b}(d, S_d \cdot x_i)) < f(\mathbf{b}(d, S_d \cdot p_i))$
(7) $S_d \cdot p_i = S_d \cdot x_i$
(8) **if** $f(\mathbf{b}(d, S_d \cdot p_i)) < f(\mathbf{b}(d, S_d \cdot p_n))$
(9) $S_d \cdot p_n = S_d \cdot p_i$
(10) **end for**
(11) Perform velocity and position updates using (2) for each particle in S_d.
(12) **end for**
(13) **Until:** stopping condition is met
The dth swarm is denoted as S_d, $S_d \cdot x_i$ denotes the current position of the ith particle in the dth swarm, $S_d \cdot p_n$ represents the best position in dth swarm. Similarly, $S_d \cdot p_i$ represents the best previous position of particle i in dth swarm. The context vector is the concatenation of $S_1 \cdot p_n, S_2 \cdot p_n, \ldots, S_D \cdot p_n$.

Algorithm 1: Pseudocode for the generic CPSO algorithm.

Table 1: Best combination levels of Sphere function by using OED method.

	x_1	x_2	x_3	Function value
C_1	(X_{L1}) 2	(X_{L1}) 0	(X_{L1}) 1	$F_1 = 5$
C_2	(X_{L1}) 2	(X_{L2}) −2	(X_{L2}) 0	$F_2 = 8$
C_3	(X_{L2}) 3	(X_{L1}) 0	(X_{L2}) 0	$F_3 = 9$
C_4	(X_{L2}) 3	(X_{L2}) −2	(X_{L1}) 1	$F_4 = 14$
Levels		Factor analysis		
L_1	$S_{11} = (F_1 + F_2)/2 = 6.5$	$S_{21} = (F_1 + F_3)/2 = 7$	$S_{31} = (F_1 + F_4)/2 = 9.5$	
L_2	$S_{12} = (F_3 + F_4)/2 = 11.5$	$S_{22} = (F_2 + F_4)/2 = 11$	$S_{32} = (F_2 + F_3)/2 = 8.5$	
L_3	(X_{L1}) 2	(X_{L1}) 0	(X_{L2}) 0	$F_{\text{final}} = 4$

values, the independent vectors X_{L1} and X_{L2}, containing good information for optimum search, will fall into different locally optimal vectors. It is desperately needed to find a method of extracting good information from X_{L1} and X_{L2} and form a new vector X_{Ln} which is closer to globally optimal value. If we exhaustively test all the combinations of X_{L1} and X_{L2} for the new vector X_{Ln}, there are 2^D trials which are unrealistic to accomplish in practice. With the help of OED [15, 16, 23, 24], a relatively good vector X_{Ln} is obtained from X_{L1} and X_{L2}, using only a few experimental tests.

The OED with both the orthogonal array (OA) and the factor analysis makes it possible to discover the best combinations for different factors with only small number of experimental samples [23, 24]. Here we have a brief discussion about OED combined with an example of vectors X_{L1} and X_{L2}. In order to simplify the problem, assuming $D = 3$, the optimization problem is to minimize the Sphere function $f(X) = x_1^2 + x_2^2 + x_3^2$, with $X_{L1} = [2, 0, 1]$, $X_{L2} = [3, -2, 0]$ which are derived from First1D and Second1D (actually First1D and Second1D should obtain the same globally optimal vector because the test function is simple; here we just want to explain the OED theory). The whole analysis of using OED for Sphere function is shown in Table 1.

OA is a predefined table which has numbers arranged in rows and columns. In Table 1, we can see that there are three factors x_1, x_2, and x_3 that affect function values and two levels X_{L1} and X_{L2} to choose. The rows represent the levels of factors in each combination, and the columns represent specific factors that can be changed from each combination [15, 24]. The programming of generating OA can be found in [24]. Factor analysis evaluates the effects of individual factors based on function values and determines the best level for each factor to form a new vector X_{Ln}. Assuming F_t denotes a function value of the combination t, where $t = 1, \ldots, n$ stands for total number of combinations. The main effect of factor d with level l as S_{dl} is expressed as

$$s_{dl} = \frac{\sum_{t=1}^{n} F_t \cdot C_{dlt}}{\sum_{t=1}^{n} C_{dlt}}, \tag{3}$$

where $d = 1, \ldots, D$ and $l = 1, 2$. If the factor is d and the level is l, $C_{dlt} = 1$, otherwise 0. In Table 1, we first calculate the effect S_{11} of level 1 in factor x_1; then the level 2 in factor x_1 is S_{12}. Since this is a minimal optimization problem, the better level 1 is chosen because the effect value of level 1 S_{11} is less than that of S_{12} in Table 1. As the example shown in Table 1, although this new vector $X_{Ln} = [2, 0, 0]$ is not shown in the four combinations tested, the best combination can be obtained by OED method.

OED works well when the interacting of vector elements is small. When the OED method is used to process some

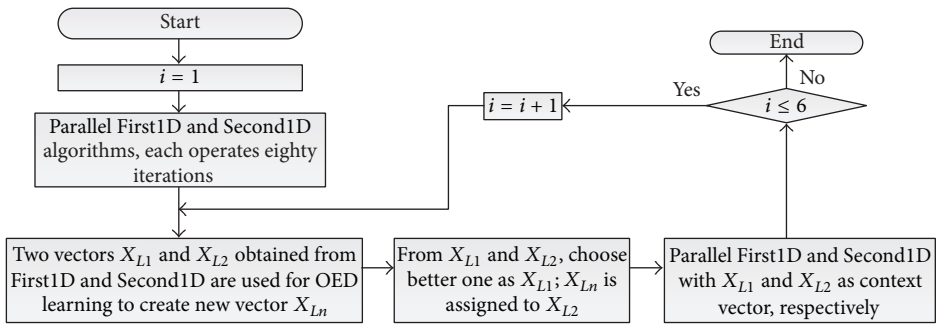

Figure 1: Flowchart of the PCPSO algorithm.

functions with high interacting vector elements, there exists a problem that the function value of new vector X_{Ln} is bigger than the function values of vector X_{L1} and X_{L2}. However, there are still two advantages by using OED in high correlated problems. One is the particle with new vector X_{Ln} jumping out of locally optimal vector; the other one is that each element of the new vector X_{Ln} chosen between X_{L1} and X_{L2} is probably closer to the globally optimal vector. Therefore, we try to use the new vector X_{Ln} as context vector to repeat the CPSO algorithm.

Based on the analysis mentioned above, we propose the PCPSO algorithm to overcome the problem of falling into local optimum. Figure 1 shows the flowchart of PCPSO. First we use two parallel CPSO algorithms (First1D and Second1D); then OED learning is applied to form the new vector X_{Ln} which is assigned to X_{L2}; the better vector chosen from X_{L1} and X_{L2} is assigned to the new X_{L1}. The new vectors X_{L1} and X_{L2}, which are treated as context vectors in CPSO algorithm, keep the search going further. In order to illustrate the efficiency of PCPSO algorithm, Figure 2 shows the convergence characteristic of the PCPSO on Rotated Ackley's function. The detailed description of Rotated Ackley's function can be found in Section 3. The parameter settings for First1D and Second1D are the same as CPSO_H algorithm [19], where ω decreases linearly from 0.9 to 0.4, $c_1 = c_2 = 1.494$, $M = 10$ stands for particle number, and $D = 30$ represents dimensions. Assuming First1D and Second1D need 80 CPSO iterations to accomplish locally optimal vectors followed with OED computation, there are 6 iterations of OED computation and therefore the total CPSO iteration is $6 \times 80 = 480$. From Figure 2, we can see that the function value drops sharply at first 80 iterations and that First1D and Second1D have similar performance. For the next 80 iterations, the better vector chosen from X_{L1} and X_{L2} is assigned to X_{L1} for another First1D implementation; the new vector X_{Ln} is assigned to X_{L2} for another Second1D implementation. The performance on First1D and Second1D are significantly different in iterations from 80 to 240. First1D algorithm gets trapped in locally optimal vector and Second1D algorithm still makes the function value drop sharply even though it is higher at start point at the 81st iteration. Since First1D algorithm has almost lost searching ability after 80 iterations, we can also adopt the strategy that Firt1D algorithm is only implemented at first 80 iterations

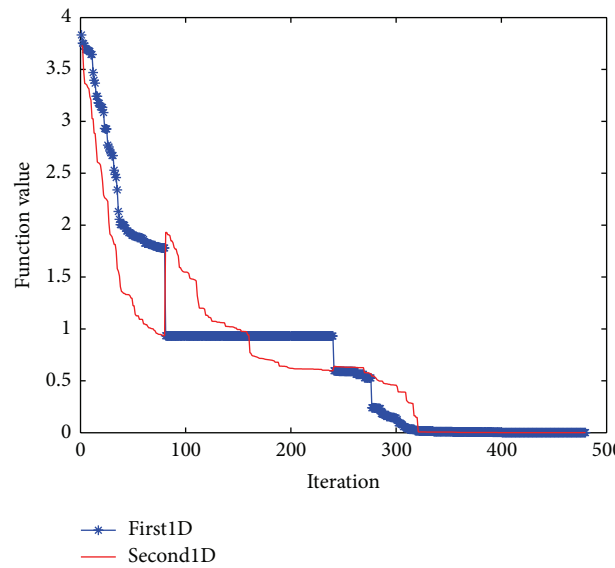

Figure 2: The convergence characteristic of the PCPSO on Rotated Ackley's function in 30 dimensions. First1D and Second1D are CPSO algorithms.

in order to save computational cost. Vector X_{L1} is used for saving the best optimal vector ever found and is combined with vector X_{L2} for new vector X_{Ln} by OED method after 80 iterations. These operations iterate generation by generation until some criteria are met.

Compared with other PSO algorithms using OED [15, 16], the PCPSO needs some extra computations of updating vector X_{L1}. However, the cooperation between local vector X_{L1} and X_{L2} makes the searching more effective. In [15], the OED result is set as an exemplar for the other particles to learn. From Figure 2, we can see that sometimes the function values after OED are even bigger. Another limitation of [15] is that the searching method gets trapped into local optimum easily because the two vectors for OED learning are similar. Based on the principle that each element of function vector moves closer to the globally optimal vector after OED, it can be seen that the new vector X_{Ln} jumps out of local optimum and keeps searching further even though the function value is higher than corresponding local optimum at start iteration.

3. Experimental Results and Discussions

3.1. Test Functions. Test functions including unimodal functions and multimodal functions are used to investigate the performance of PSO algorithms [25]. We choose 15 test functions on 30 dimensions. The formulas and the properties of these functions are presented below.

Group A: unimodal and simple multimodal functions are as follows:

(1) Sphere function

$$f_1(x) = \sum_{i=1}^{D} x_i^2, \tag{4}$$

(2) Rosenbrock's function

$$f_2(x) = \sum_{i=1}^{D-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2, \tag{5}$$

(3) Griewank's function

$$f_3(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{6}$$

(4) Rastrigin's function

$$f_4(x) = x_i^2 - 10\cos(2\pi x_i) + 10, \tag{7}$$

(5) Ackley's function

$$\begin{aligned} f_5(x) = & -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i-1}^{D} x_i^2}\right) \\ & - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e, \end{aligned} \tag{8}$$

(6) Weierstrass function

$$\begin{aligned} f_6(x) = & \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k\left(x_i + 0.5\right)\right)\right]\right) \\ & - D\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(\pi b^k\right)\right] \\ & a = 0.5, \quad b = 3, \quad k_{\max} = 20, \end{aligned} \tag{9}$$

(7) Noncontinuous Rastrigin's function

$$\begin{aligned} f_7(x) = & \sum_{i=1}^{D}\left(y_i^2 - 10\cos(2\pi y_i) + 10\right), \\ y_i = & \begin{cases} x_i & |x_i| < \frac{1}{2} \\ \dfrac{\text{round}(2x_i)}{2} & |x_i| \geq \frac{1}{2} \end{cases} \\ & \text{for } i = 1, 2, \ldots, D, \end{aligned} \tag{10}$$

(8) Schwefel's function

$$f_8(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin\left(\left|x_i\right|^{1/2}\right). \tag{11}$$

Group B: Rotated and shifted multimodal functions are as follows.

In Group A, the functions can be divided into smaller search spaces because of low interacting of the elements. Therefore, the CPSO algorithm is suitable to optimize these functions. In Group B, seven rotated multimodal functions are generated by an orthogonal matrix M to test the performance of the PCPSO algorithm. The new rotated vector $y = M * x$, which is obtained through the original vector x left multiplied by orthogonal matrix M, performs like the high interacting vector, because all elements in vector y will be changed once one element in vector x changes. Detailed illustration of generating the orthogonal matrix is introduced in [26]. Meanwhile, one shifted and rotated function is also listed in group B to test the performance of PSO algorithm. Table 2 shows the globally optimal vectors x^*, the corresponding function values $f(x^*)$, the search range, and the initialization range of each function. Consider the following:

(9) Rotated Ackley's function

$$\begin{aligned} f_9(x) = & -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) \\ & - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi y_i)\right) + 20 + e, \quad y = M * x, \end{aligned} \tag{12}$$

(10) Rotated Rastrigin's function

$$f_{10}(x) = \sum_{i=1}^{D} y_i^2 - 10\cos(2\pi y_i) + 10, \quad y = M * x, \tag{13}$$

(11) Rotated Griewank's function

$$f_{11}(x) = \sum_{i=1}^{D} \frac{y_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{y_i}{\sqrt{i}}\right) + 1, \quad y = M * x, \tag{14}$$

(12) Rotated Weierstrass function

$$\begin{aligned} f_{12}(x) = & \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k\left(y_i + 0.5\right)\right)\right]\right) \\ & - D\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(\pi b^k\right)\right] \\ & a = 0.5, \quad b = 3, \quad k_{\max} = 20, \quad y = M * x, \end{aligned} \tag{15}$$

Table 2: Globally optimal vector x^*, corresponding function value $f(x^*)$, search range, and initialization range of test functions.

Functions	x^*	$f(x^*)$	Search range	Initialization range
f_1	$[0, 0, \ldots, 0]$	0	$[-100, 100]^D$	$[-100, 50]^D$
f_2	$[1, 1, \ldots, 1]$	0	$[-2, 2]^D$	$[-2, 2]^D$
f_3	$[0, 0, \ldots, 0]$	0	$[-600, 600]^D$	$[-600, 200]^D$
f_4	$[0, 0, \ldots, 0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_5	$[0, 0, \ldots, 0]$	0	$[-32, 32]^D$	$[-32, 16]^D$
f_6	$[0, 0, \ldots, 0]$	0	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$
f_7	$[0, 0, \ldots, 0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_8	$[420.96, 420.96, \ldots, 420.96]$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_9	$[0, 0, \ldots, 0]$	0	$[-32, 32]^D$	$[-32, 16]^D$
f_{10}	$[0, 0, \ldots, 0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_{11}	$[0, 0, \ldots, 0]$	0	$[-600, 600]^D$	$[-600, 200]^D$
f_{12}	$[0, 0, \ldots, 0]$	0	$[-0.5, 0.5]^D$	$[-0.5, 0.2]^D$
f_{13}	$[0, 0, \ldots, 0]$	0	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$
f_{14}	$[420.96, 420.96, \ldots, 420.96]$	0	$[-500, 500]^D$	$[-500, 500]^D$
f_{15}	$[o_1, o_2, \ldots, o_D]$	−330	$[-5.12, 5.12]^D$	$[-5.12, 2]^D$

(13) Rotated noncontinuous Rastrigin's function

$$f_{13}(x) = \sum_{i=1}^{D} z_i^2 - 10\cos(2\pi z_i) + 10,$$

$$z_i = \begin{cases} y_i & |y_i| < \frac{1}{2} \\ \dfrac{\text{round}(2y_i)}{2} & |y_i| \geq \frac{1}{2} \end{cases} \quad (16)$$

$$\text{for } i = 1, 2, \ldots, D, \quad y = M * x,$$

(14) Rotated Schwefel's function

$$f_{14}(x) = 418.9829 \times D - \sum_{i=1}^{D} z_i,$$

$$z_i = \begin{cases} y_i \sin\left(|y_i|^{1/2}\right) & \text{if } |y_i| \leq 500 \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

$$\text{for } i = 1, 2, \ldots, D,$$

$$y = y' + 420.96, \qquad y' = M * (x - 420.96),$$

(15) Rotated and Shifted Rastrigin's function

$$f_{15}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right) - 330, \quad (18)$$

$$z = M * y, \quad y = x - o, \quad o = [o_1, o_2, \ldots, o_D],$$

where $o = [o_1, o_2, \ldots, o_D]$ is the shifted globally optimal vector.

3.2. PCPSO's Performance on 15 Functions. The performance of PCPSO is tested by functions f_1–f_{15} which includes multimodal, rotated, and shifted functions in 30 dimensions. The parameter settings for the PCPSO algorithm are as follows: ω decreases linearly from 0.9 to 0.4, $c_1 = c_2 = 1.494$, and particle number $M = 10$. Since the vectors of f_1–f_8 have very low interacting elements, Figure 3 shows the convergence characteristics of PCPSO algorithm on test functions f_1–f_8 in 30 iterations, where the y-axis is expressed in log format. From Figure 3, we can see that all the function values drop significantly in 30 iterations except Rosenbrock's function. In order to fully test PCPSO's performance on Rosenbrock's function, we make a comparison between PCPSO and CPSO algorithm in Figure 4. The upper plot of Figure 4 shows convergence characteristic of PCPSO which cooperates with

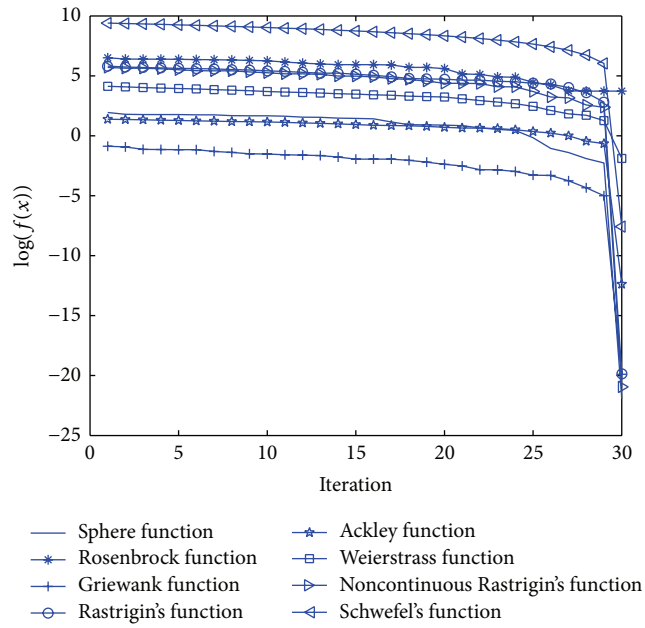

Figure 3: The convergence characteristics of the PCPSO algorithm on test functions f_1–f_8 in 30 iterations.

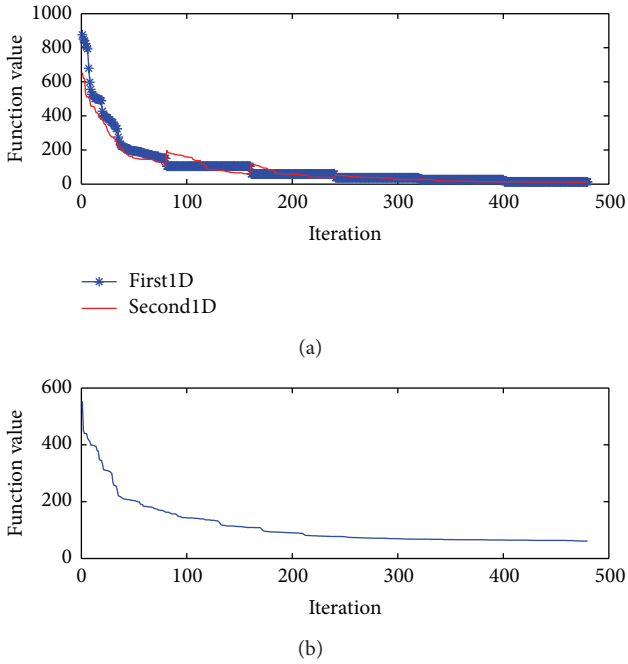

Figure 4: The convergence characteristics of PCPSO and CPSO algorithms on Rosenbrock's function. (a) PCPSO, (b) CPSO.

the two independent algorithms (First1D and Second1D) by using OED learning; the lower plot is the convergence characteristic of CPSO algorithm. From Figure 4, we can see that the OED learning makes the function vector jump out of locally optimal vector and drops the function value finally. The convergence characteristics for the remaining six test functions f_{10}–f_{15} are shown in Figure 5. All these six functions are rotated multimodal and nonseparable functions. Similar to Figure 2, in spite the fact that the function value of Second1D is increased in some cases at the start point caused by OED learning, it will drop sharply with the increase of iterations. This phenomenon is especially obvious in test function f_{12}. To sum up, PCPSO works well in terms of preventing locally optimal vector and dropping function value sharply. The OED learning finds the best combination of two locally optimal vectors X_{L1} and X_{L2} in iterations 80, 160, and 240. However, there are still some drawbacks causing the vector falling into locally optimal vector like f_{14}; the reason is that the vectors X_{L1} and X_{L2} are equal to each other, the OED learning ability is lost finally.

3.3. Comparison with Other PSOs and Discussions. Seven PSO variants are taken from the literature for comparison on 15 test functions with 30 dimensions. In the following, we briefly describe the peer algorithms shown in Table 3. The first algorithm is the standard PSO (SPSO), whose performance is improved by using a random topology. The second algorithm fully informed PSO (FIPS) uses all the neighbor particles to influence the flying velocity. The third orthogonal PSO (OPSO) aims to generate a better position by using OED. The fourth algorithm fitness-distance-ratio PSO (FDR_PSO) solves the premature convergence problem

Table 3: Parameter settings for the seven PSO variants.

Algorithm	Parameter settings
SPSO [27]	ω: 0.4~0.9, $c_1 = c_2 = 1.193$
FIPS [10]	$\chi = 0.7298$, $\sum c_i = 4.1$
OPSO [16]	ω: 0.4~0.9, $c_1 = c_2 = 2.0$
FDR_PSO [13]	ω: 0.4~0.9, $\sum c_i = 4.0$
CLPSO [6]	ω: 0.4~0.9, $c_1 = c_2 = 2.0$
CPSO_H [19]	ω: 0.4~0.9, $c = 1.49$, $\kappa = 6$
OLPSO [15]	ω: 0.4~0.9, $c = 2.0$, $G = 5$

by using the fitness distance ratio. The fifth comprehensive learning PSO (CLPSO) is proposed for solving multimodal problems. The sixth cooperative PSO (CPSO_H) uses one-dimensional swarms to search each dimension separately. The seventh orthogonal learning PSO (OLPSO) can guide particles to fly towards an exemplar which is constructed by P_i and P_n using OED method.

The swarm size is set 40 for the seven PSO variants mentioned above and 10 for the PCPSO algorithm. The maximal FEs for the eight algorithms are set 200,000 in each run of each test function. All functions were run 25 times and the mean values and standard deviation of the results are presented in Table 4. The best results are shown in boldface. From the results, we observe that CPSO_H, CLPSO, and OLPSO perform well on the first 8 functions. The reason is that CPSO-H suits well on 1-dimensional search, CLPSO and OLPSO can search further by using their own learning strategies. The learning strategy in CLPSO is to prevent falling into local optimum by random learning, while in OLPSO the strategy is to find a better combination of locally optimal vector. PCPSO method can keep searching by the cooperation of two locally optimal vectors X_{L1} and X_{L2}. With the implementation of Second1D, the algorithm can always find a better vector X_{L2} compared with the previous best one X_{L1} by using OED method. The PCPSO method performs well on f_2-f_3, f_9–f_{11}, f_{13}, and f_{15}. However, the function value obtained by this method is not close enough to the globally optimal value, the premature convergence still happens in f_{12} and f_{14} due to lack of vector diversity. The advantages of this method are the rapid convergent speed and robustness; whatever test functions are, the algorithm can find an acceptable optimum especially when facing some complex functions. The nonparametric Wilcoxon rank sum tests are taken as t-test in Table 5 to determine whether the results obtained by PCPSO are statistically different from the best results achieved by the other algorithms. A value of one indicates that the performance of the two algorithms is statistically different with 95% certainty, whereas the value of zero implies that the performance is not statistically different. Among these 15 functions, there are 10 functions that are statistically different between the best results and the results of PCPSO. From Tables 4 and 5, we can see that the PCPSO obtains 5 best functions (f_2, f_3, f_{10}, f_{11}, f_{13}) that are statistically different from the others. In addition, most of the best functions obtained by PCPSO are focused on

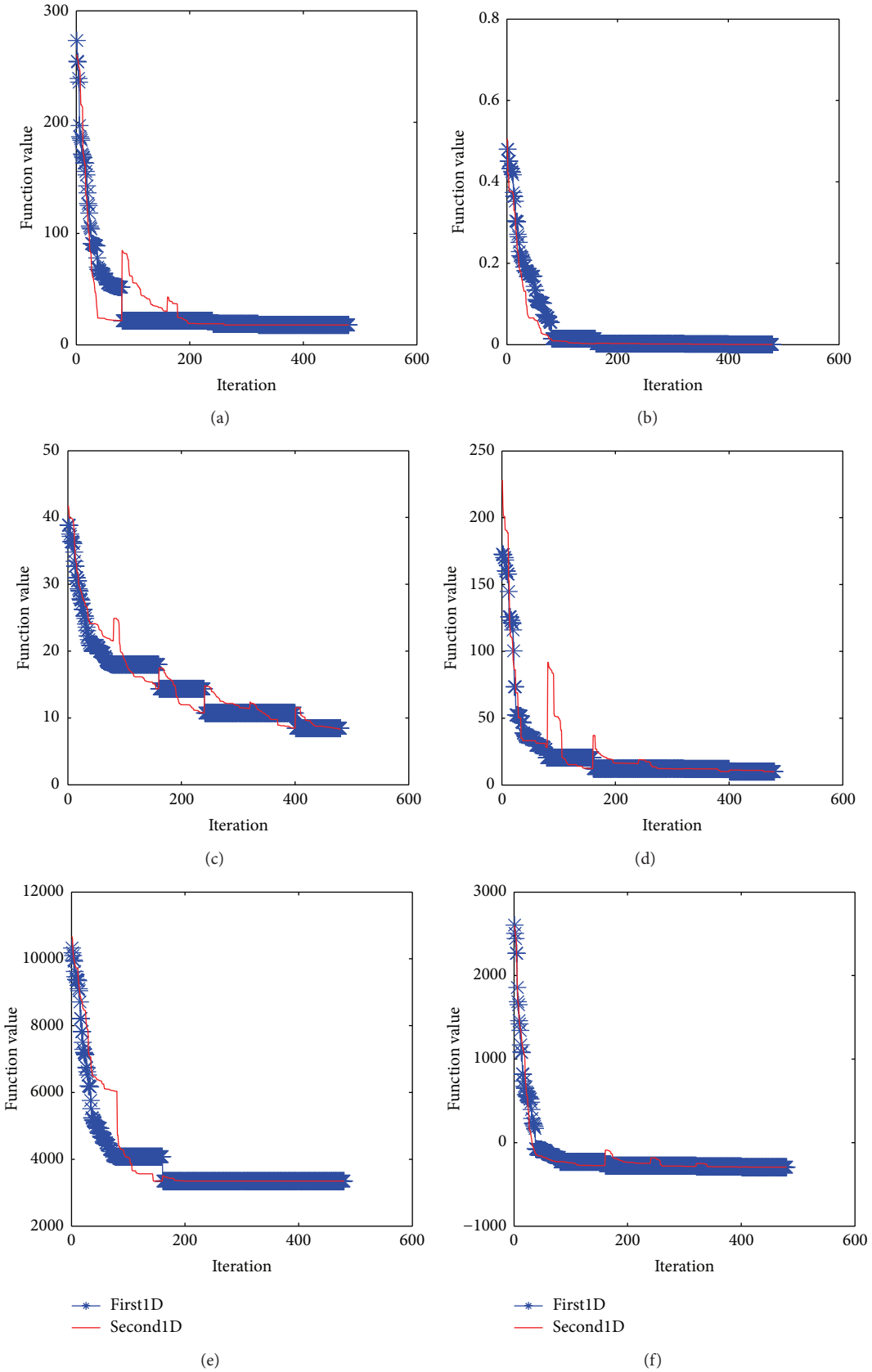

FIGURE 5: The convergence progress of 30-dimensional test functions using PCPSO. (a) Rotated Rastrigin's function, (b) Rotated Griewank's function, (c) Rotated Weierstrass function, (d) Rotated noncontinuous Rastrigin's function, (e) Rotated Schwefel's function, and (f) Rotated and Shifted Rastrigin's function.

Table 4: Results for 30D functions.

F	SPSO	FIPS	OPSO	FDR_PSO
f_1	$1.16e-069 \pm 2.86e-068$	$2.69e-012 \pm 2.28e-011$	$6.71e-018 \pm 6.82e-018$	$\mathbf{4.88e-102 \pm 1.53e-101}$
f_2	$1.68e+001 \pm 8.56e+000$	$2.47e+001 \pm 2.19e-001$	$4.12e+001 \pm 1.19e+001$	$1.54e+001 \pm 6.79e+000$
f_3	$1.73e-002 \pm 1.08e-001$	$4.65e-003 \pm 3.71e-002$	$2.10e-003 \pm 6.19e-003$	$1.72e-002 \pm 9.08e-003$
f_4	$4.61e+001 \pm 7.85e+001$	$7.44e+001 \pm 1.56e+001$	$3.27e+000 \pm 6.35e+000$	$2.91e+001 \pm 8.06e+000$
f_5	$1.27e+000 \pm 4.79e+000$	$5.12e-007 \pm 7.81e-007$	$3.74e-009 \pm 5.96e-009$	$2.85e-014 \pm 4.32e-015$
f_6	$2.38e+000 \pm 8.04e+000$	$3.13e-001 \pm 3.74e-001$	$3.28e-002 \pm 4.93e-002$	$7.75e-003 \pm 1.21e-002$
f_7	$5.21e+001 \pm 4.19e+001$	$6.38e+001 \pm 9.05e+000$	$4.23e+000 \pm 8.24e+000$	$1.45e+001 \pm 6.58e+000$
f_8	$3.76e+003 \pm 3.43e+003$	$2.56e+003 \pm 7.31e+002$	$2.72e+003 \pm 3.31e+003$	$3.16e+003 \pm 4.65e+002$
f_9	$1.35e+000 \pm 1.02e+000$	$7.26e-005 \pm 1.42e-005$	$4.07e-005 \pm 8.26e-006$	$3.63e-001 \pm 5.72e-001$
f_{10}	$8.38e+001 \pm 1.53e+002$	$1.75e+002 \pm 5.87e+001$	$1.64e+002 \pm 3.98e+001$	$4.67e+001 \pm 1.44e+001$
f_{11}	$1.16e-002 \pm 1.28e-002$	$6.95e-005 \pm 3.21e-005$	$1.13e-003 \pm 2.45e-003$	$8.79e-003 \pm 1.07e-002$
f_{12}	$2.83e+001 \pm 5.31e+000$	$1.35e+001 \pm 8.39e+000$	$1.21e+001 \pm 1.05e+001$	$2.35e+000 \pm 1.41e+000$
f_{13}	$5.51e+001 \pm 2.07e+001$	$8.74e+001 \pm 1.92e+001$	$7.67e+001 \pm 7.43e+001$	$4.64e+001 \pm 9.01e+000$
f_{14}	$5.59e+003 \pm 6.07e+002$	$2.67e+003 \pm 7.72e+002$	$2.45e+003 \pm 2.89e+002$	$3.80e+003 \pm 6.18e+002$
f_{15}	$-2.49e+002 \pm 4.76e+001$	$-1.07e+002 \pm 1.52e+002$	$-2.26e+002 \pm 1.07e+001$	$-2.71e+002 \pm 3.69e+001$
f	CLPSO	CPSO_H	OLPSO	PCPSO
f_1	$4.46e-014 \pm 1.51e-014$	$1.16e-033 \pm 2.26e-033$	$1.16e-038 \pm 1.56e-038$	$1.12e-030 \pm 5.14e-31$
f_2	$2.10e+001 \pm 2.71e+000$	$2.06e+001 \pm 1.13e+001$	$1.74e+001 \pm 1.05e+001$	$\mathbf{1.12e+001 \pm 8.19e+000}$
f_3	$3.14e-010 \pm 4.64e-010$	$3.63e-002 \pm 7.11e-002$	$3.17e-010 \pm 6.23e-011$	$\mathbf{3.36e-012 \pm 7.51e-013}$
f_4	$4.76e-010 \pm 2.17e-010$	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$	$2.21e-012 \pm 6.14e-013$
f_5	$4.97e-014 \pm 2.12e-014$	$4.56e-014 \pm 1.25e-014$	$\mathbf{4.76e-015 \pm 1.57e-016}$	$4.83e-008 \pm 1.38e-008$
f_6	$4.34e-007 \pm 2.09e-007$	$\mathbf{8.15e-015 \pm 8.92e-016}$	$1.27e-002 \pm 2.79e-002$	$2.41e-011 \pm 1.09e-011$
f_7	$\mathbf{4.36e-010 \pm 2.44e-010}$	$1.21e-001 \pm 3.19e-001$	$4.51e-008 \pm 1.17e-008$	$2.65e-007 \pm 2.27e-007$
f_8	$\mathbf{1.26e-012 \pm 8.62e-013}$	$1.08e+003 \pm 2.27e+002$	$3.88e-004 \pm 6.21e-005$	$3.82e-004 \pm 7.31e-006$
f_9	$3.47e-004 \pm 2.11e-004$	$2.13e+000 \pm 3.55e-001$	$2.28e-005 \pm 1.14e-005$	$\mathbf{2.25e-006 \pm 1.08e-006}$
f_{10}	$3.35e+001 \pm 5.58e+000$	$8.04e+001 \pm 2.23e+001$	$5.77e+001 \pm 1.27e+001$	$\mathbf{1.63e+000 \pm 5.18e+000}$
f_{11}	$7.07e-005 \pm 1.18e-005$	$5.54e-002 \pm 4.79e-002$	$2.73e-005 \pm 3.08e-005$	$\mathbf{1.22e-005 \pm 5.15e-006}$
f_{12}	$\mathbf{3.09e+000 \pm 1.51e+000}$	$1.53e+001 \pm 3.62e+000$	$7.03e+000 \pm 1.41e+000$	$9.37e+000 \pm 3.58e+000$
f_{13}	$3.53e+001 \pm 5.52e+000$	$8.60e+001 \pm 2.61e+001$	$4.03e+001 \pm 1.34e+001$	$\mathbf{1.07e+001 \pm 2.37e+000}$
f_{14}	$\mathbf{2.46e+003 \pm 1.88e+003}$	$3.82e+003 \pm 6.62e+002$	$3.15e+003 \pm 1.27e+003$	$3.26e+003 \pm 1.23e+003$
f_{15}	$-2.97e+002 \pm 2.13e+001$	$-2.38e+002 \pm 7.42e+001$	$-2.76e+002 \pm 3.47e+001$	$\mathbf{-3.08e+002 \pm 1.02e+001}$

Table 5: t-test on 15 functions.

Functions	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
t-test	1	1	1	1	1	0	0	1
Functions	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	
t-test	0	1	1	1	1	0	0	

nonseparable multimodal problems, which demonstrates the effectiveness and robustness of PCPSO algorithm.

4. Conclusion

In this paper, a novel PCPSO algorithm is presented in order to overcome the problem of falling into local optimum. In PCPSO, considering the complex interacting of the elements of the vector, the CPSO (First1D and Second1D) algorithm is used twice to create two locally optimal vectors. These two vectors are combined together by OED learning so that the best elements can be chosen to jump out of local optimum and search further.

Although OED operation may cause the function value higher than local optima in some cases, the function value will always drop sharply after OED because the OED operation chooses the elements that are closer to globally optimal vector. Experimental tests have been conducted on 15 benchmark functions including unimodal, multimodal, rotated, and shifted functions. From the comparative results, it can be concluded that PCPSO significantly improves the performance of PSO and provides a better solution on most rotated multimodal problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (51275353), Macao Science and Technology Development Fund (108/2012/A3, 110/2013/A3), and Research Committee of University of Macau (MYRG183(Y1-L3)FST11-LYM, MYRG203(Y1-L4)-FST11-LYM).

References

[1] R. C. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, October 1995.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[3] A. Adam, M. I. Shapiai, M. Z. M. Tumari, M. S. Mohamad, and M. Mubin, "Feature selection and classifier parameters estimation for EEG signals peak detection using particle swarm optimization," *The Scientific World Journal*, vol. 2014, Article ID 973063, 13 pages, 2014.

[4] N. Geng, D. W. Gong, and Y. Zhang, "PSO-based robot path planning for multisurvivor rescue in limited survival time," *Mathematical Problems in Engineering*, vol. 2014, Article ID 187370, 10 pages, 2014.

[5] M. Shoaib and W.-C. Song, "Data aggregation for Vehicular Adhoc Network using particle swarm optimization," in *Proceedings of the 14th Asia-Pacific Network Operations and Management Symposium*, pp. 1–6, September 2012.

[6] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[7] Y. H. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 591–600, Springer, Berlin, Germany, 1998.

[8] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 3, pp. 627–646, 2012.

[9] X. Li, "Niching without niching parameters: particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, 2010.

[10] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[11] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, pp. 1671–1676, Honolulu, Hawaii, USA, May 2002.

[12] T. M. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[13] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 174–181, IEEE, April 2003.

[14] X. Chen and Y. M. Li, "A modified PSO structure resulting in high exploration ability with convergence guaranteed," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 37, no. 5, pp. 1271–1289, 2007.

[15] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[16] S. Y. Ho, H. S. Lin, W. H. Liauh, and S. J. Ho, "OPSO: Orthogonal particle swarm optimization and its application to task assignment problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 2, pp. 288–298, 2008.

[17] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[18] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature—PPSN III*, vol. 866 of *Lecture Notes in Computer Science*, pp. 249–257, Springer, Berlin, Germany, 1994.

[19] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[20] G. Zhang and Y. M. Li, "Cooperative particle swarm optimizer with elimination mechanism for global optimization of multimodal problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 210–217, Beijing, China, July 2014.

[21] G. Zhang and Y. M. Li, "Orthogonal experimental design method used in particle swarm optimization for multimodal problems," in *Proceedings of the 6th International Conference on Advanced Computational Intelligence (ICACI '13)*, pp. 183–188, Hangzhou, China, October 2013.

[22] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[23] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, New York, NY, USA, 5th edition, 2000.

[24] S.-Y. Ho, L.-S. Shu, and J.-H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 6, pp. 522–541, 2004.

[25] X. Yao, Y. Liu, and G. M. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[26] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms," *BioSystems*, vol. 39, no. 3, pp. 263–278, 1996.

[27] SPSO, 2007, http://www.particleswarm.info/Programs.html.