2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Cooperative Particle Swarm Optimizer with Elimination Mechanism for Global Optimization of Multimodal Problems

Geng Zhang[1]
[1]*Department of Electromechanical Engineering*
University of Macau, Taipa, Macao SAR, China

Yangmin Li[1,2]
[2]*School of Mechanical Engineering*
Tianjin University of Technology
Tianjin 300191, China
Corresponding Author: E-mail: ymli@umac.mo

***Abstract*—This paper presents a new particle swarm optimizer (PSO) that called the cooperative particle swarm optimizer with elimination mechanism (CPSO-EM) in an attempt to address the issue of getting trapped into local optimum when solving nonseparable multimodal problems using PSO algorithm. The proposed CPSO-EM builds on the basis of an early cooperative PSO (CPSO-H) that employs cooperative behavior. The CPSO-H and elimination mechanism (EM) memory are incorporated together to obtain CPSO-EM. Experimental studies on a set of test functions show that CPSO-EM exhibits better performance in solving nonseparable multimodal problems than several other peer algorithms.**

I. Introduction

In the past decade, particle swarm optimizer (PSO) has been applied and studied by many researchers with promising results [1]. However, it may fall into local optimum more easily as the global optimization benchmark problems become more complex. Better optimization algorithms are always needed for solving more and more 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:

$$Minf(X), \quad X = [x_1, x_2, ..x_d.., x_D] \tag{1}$$

where $X = [x_1, x_2, ..x_d.., x_D]$ is the vector to be optimized and D is the number of parameters [2].

PSO is notorious for being prone to premature convergence. Among the PSO variants in the literature, one of the main methods to prevent premature convergence is to increase the diversity of particles which represent potential solutions in PSO [2]-[5]. However, the performance is not always satisfactory due to the large search space on high dimensional multimodal problems. Meanwhile, some cooperative coevolutionary algorithms are presented which adopt the divide-and-conquer strategy to tackle high dimensional problems [6]-[8]. It is realized that the performance of cooperative coevolutionary algorithm deteriorates when there exists interdependencies among parameters. In order to cope with this problem, the decomposition strategy called random grouping and adaptive weighting scheme is implemented in the literature. However, the probability of optimizing interdependencies parameters is very low since the other parameters contained in the same subcomponent limit the performance. In this paper, we present a novel algorithm called cooperative particle swarm optimizer with elimination mechanism (CPSO-EM) to cope with the nonseparable problems. Based on an early cooperative PSO (CPSO-H) [6], CPSO-EM overcomes the problem of falling into local optimum by using elimination mechanism (EM) memory that increases the diversity of useful particles.

The rest of this paper is organized as follows. In Section II, basic PSO algorithm and some variants are presented. Section III describes the CPSO-EM algorithm. Section IV describes the benchmark test functions and compares the performance of CPSO-EM with that of some peer algorithms taken from literature. Final conclusions are given in Section V.

This work was supported by Macao Science and Technology Development Fund (108/2012/A3), Research Committee of University of Macau under Grant No.:MYRG183(Y1-L3)FST11-LYM and MYRG203(Y1-L4)-FST11-LYM).

II. PSO and Some Variants

A. Basic PSO algorithm

Inspired from social behavior and cognitive behavior, Kennedy and Eberhart presented PSO algorithm to search for optimal values through population-based iterative learning algorithm [9],[10]. A particle which represents a potential solution is a point in the D-dimensional search space. Let M denote the size of the swarm, the current state of each particle i is represented by its position vector $X_i = [x_{i1}, x_{i2}, ..x_{id}.., x_{iD}]$ and the movement of particle i is represented by velocity vector $V_i = [v_{i1}, v_{i2}, ..v_{id}.., v_{iD}]$, where $i = 1, 2, ..., M$ is positive integer indexing particle in the swarm. Using $t = 1, 2, ..., T$ represents the iteration number, the velocity $v_{id}(t)$ and the position $x_{id}(t)$ can be updated as follows:

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

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{3}$$

where the inertia weight variable $\omega \in (0, 1)$ 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. Therefore, the inertia weight ω sometimes decreases linearly from 0.9 to 0.4 during the operation [11]. c_1 and c_2 denote the acceleration constants which are usually set 2.0 or adaptively controlled by the evolutionary states [12] [13]. It is shown that sometimes assigning different values to c_1 and c_2 may lead to better performance [14]. $r1_{id}$ and $r2_{id}$ are random numbers generated between

0 and 1 for the dth dimension of ith particle. $P_i(t) = [p_{i1}(t), p_{i2}(t), ..p_{id}(t).., p_{iD}(t)]$ represents the best previous position of particle i which is defined by $pbest$ from the previous t iterations, $P_n(t) = [p_{n1}(t), p_{n2}(t), ..p_{nd}(t).., p_{nD}(t)]$ represents the best position among particle i's neighborhood which is defined by $gbest$ (global best) or $lbest$ (local best) from the previous t iterations. The position vector $X_i = [x_{i1}, x_{i2}, ..x_{id}.., x_{iD}]$ and the velocity vector $V_i = [v_{i1}, v_{i2}, ..v_{id}.., v_{iD}]$ are initialized randomly and are updated by (2) and (3) generation by generation until some criterions are met.

B. Some PSO Variants

Many PSO variants have been reported with enhanced searching performance in literature since the PSO method was first introduced by Kennedy and Eberhart. Adaption is one of the research trend in PSO variants. In [16], a PSO version called adaptive PSO (APSO) with adaptive ω, c_1 and c_2 is proposed where there are four evolutionary states, namely, "exploitation", "exploration","convergence", and "jumping out". In each evolutionary state, APSO gives one corresponding equation to adjust the value of c_1 and c_2. Meanwhile, the inertia weight ω is tuned using a sigmoid mapping. In [15], an adaptive learning method called self learning PSO (SLPSO) is proposed, which allows each particle to have a set of four strategies implemented by adaptive learning framework at the individual level to cope with different situations in the search space. In [17], a population manager method for PSO is proposed to dynamically adjust the number of particles according to some heuristic conditions.

Increasing particle diversity is another trend for PSO algorithm. Suganthan suggested a method adjusting the neighbor model dynamically by utilizing the $lbest$ model to increase the particle diversity [18]. In [19], Mendes et al. proposed a fully informed PSO (FIPS) method to weight the influence of each particle to its neighbors based on its fitness value and the neighborhood size. In [5], repulsive, collision avoiding forces were presented in order to increase particle diversity by Blackwell. He also presented a method to split the population of particles into a set of interacting swarms. These swarms interact locally by an exclusion parameter and globally through a new anti-convergence operator [3]. In [20], wavelet theory was applied to enhance the PSO in exploring the search space more effectively for a better solution. Chen and Li proposed a method to improve the searching ability by combining PSO with a controllable random exploration velocity [4]. Fitness-distance-ratio PSO (FDR-PSO) algorithm updates each velocity by selecting another particle that has better fitness value and is closer to the particle being updated [21]. In [2], a learning algorithm called comprehensive learning PSO (CLPSO) was proposed where a particle used different particles' historical best information to update its velocity.

Hybridization of PSO with other search techniques and cooperative coevolving multi-swarm have also been applied to improve the performance of PSO algorithm. Hybridization of PSO with genetic algorithm (GA) theory was first introduced by Angeline who put a selection operator into PSO to improve the performance of original PSO [22]. A hybrid of GA and PSO called HGAPSO was proposed by C.F.Juang [23], in which individuals in new generations were created by the combination of crossover and mutation operation with PSO. Meanwhile, there are some cooperative coevolving multi-swarm like cooperative PSO (CPSO-H$_k$) which uses several subcomponent swarms to search each subcomponent separately and is combined with the original PSO to improve the performance on multimodal problems [6]. In [24], the dynamic multi-swarm PSO are combined with a sub-regional harmony search (SHS) to enhance the performance of PSO algorithm. In order to deal with the nonseparable optimization problems by cooperative coevolving multi-swarm, the algorithms using random grouping and adaptive weighting schemes were presented and analyzed in [7] and [8].

III. Cooperative Particle Swarm Optimizer with Elimination Mechanism

Although different kinds of adaption methods or increasing particle diversity methods are presented, it is very difficult to find the global optimum due to the large search space on high dimensional multimodal problems and premature convergence is still the main deficiency of PSO algorithm [25]. The cooperative coevolving method provides a very promising solution of large search space by dividing the search space into lower dimensional subspaces and has shown its effectiveness on separable problems. For nonseparable problems, random grouping and adaptive weighting schemes are presented to improve the cooperative coevolving performance [7] and [8]. However, there are other parameters except interdependency parameters contained in the random divided subspaces limit the optimization of interdependency parameters. In order to solve the nonseparable problems, we present a novel algorithm called cooperative particle swarm optimizer with elimination mechanism (CPSO-EM) which is based on a cooperative algorithm shown in [6] and the elimination mechanism (EM) memory is adopted to overcome the problem of interdependency. In this section, we first have a brief introduction of the CPSO-H algorithm in [6], then based on the features of CPSO-H algorithm, the EM memory is introduced to store useful local optimal vectors from which useful elements can be extracted.

A. CPSO-H Algorithm

Van den Bergh and Engelbrecht [6] presented two cooperative PSO algorithms: CPSO-S$_k$ and CPSO-H$_k$. CPSO-S$_k$ allows a vector to be split into K components and there are K swarms to search each component separately. CPSO-H$_k$ is a hybrid method combining a standard PSO with the CPSO-S$_k$ to overcome the problem of stagnation. In this paper, we assume that each component contains only a single dimension, therefore, a D dimensional vector is split into D swarms. Since one dimensional swarm searches each dimension separately, we name the CPSO-S$_k$ and CPSO-H$_k$ as CPSO-S and CPSO-H, respectively.

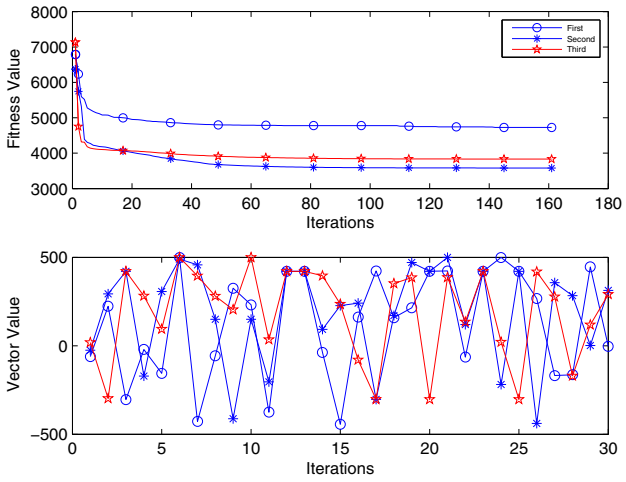

Fig. 1. CPSO-H's convergence characteristics and solution vectors on Rotated Schwefel's function. The algorithm repeats three times.

Algorithm 1 shows the pseudocode of CPSO-H algorithm. In order to evaluate the fitness of a particle in swarm S_d, a context vector p_n is required, which is the concatenation of all global best particles from D one-dimensional swarms. The ith particle in the dth swarm is evaluated by using the function $b(d, S_d.x_i)$ (shown in Algorithm 1) which has a D dimensional vector consisting of context vector p_n with its dth component replaced by $S_d.x_i$. After one iteration of one-dimensional swarms, the original PSO algorithm is followed. First, the context vector expressed as $b(1, S_1.p_n)$ in Algorithm 1 is used to overwrite a randomly chosen particle from Q swarm; then a new global best vector $Q.p_n$ is obtained with one iteration of the Q swarm. If the function value of $Q.p_n$ is better than that of context vector $b(1, S_1.p_n)$, $Q.p_n$ will be used to overwrite the position of context vector (shown as $S_d.p_n$ in Algorithm 1). In practical problems, there are bounds on the variables' ranges. Assuming the search range for a problem is $[X_{min}, X_{max}]$ where $X_{min} = [x_{1min}, x_{2min}, ..x_{dmin}.., x_{Dmin}]$ and $X_{max} = [x_{1max}, x_{2max}, ..x_{dmax}.., x_{Dmax}]$, the equation $x_{id} = min(x_{dmax}, max(x_{dmin}, x_{id}))$ is used to restrain a particle on the border once it moves out of the search range in Algorithm 1.

Since CPSO-H algorithm uses one dimensional swarm for each dimension and interdependency elements of the solution vector can not be changed simultaneously, it is easier to fall into local optimum. Therefore, the performance of CPSO-H depends on numerical distribution when particles $X = [x_1, x_2, ..x_d.., x_D]$ are initialized by random values. Among the particles, the one assigned as context vector almost determines the performance of CPSO-H. However, it has fast convergence speed because the search space is reduced significantly and is very effective on separable problems and it can still obtain global elements if the initialization of particles fall into the local search zones where global elements

Algorithm 1: Pseudocode for the generic CPSO-H algorithm.

Create and initialize D one-dimensional PSOs: S_d $d \in [1..D]$
Create and initialize an D-dimensional PSO: Q
repeat:
for each swarm $d \in [1..D]$:
 for each particle $i \in [1..M]$:
  if $f(\mathbf{b}(d, S_d.x_i)) < f(\mathbf{b}(d, S_d.p_i))$
   $S_d.p_i = S_d.x_i$
  endif
  if $f(\mathbf{b}(d, S_d.y_i)) < f(\mathbf{b}(d, S_d.p_n))$
   $S_d.p_n = S_d.p_i$
  endif
 endfor
 Perform velocity and position updates using equations (2) and (3) for each particle in S_d
 if $S_d.p_n > x_{dmax} | S_d.p_n < x_{dmin}$
  $S_d.p_n = min(x_{dmax}, max(x_{dmin}, S_d.p_n))$
 endif
endfor
Select random $k \sim U(1, M/2) | Q.p_k \neq Q.p_n$
$Q.x_k = \mathbf{b}(1, S_1.p_n)$
for each particle $i \in [1..M]$:
 if $f(Q.x_i) < f(Q.p_i)$
  $Q.p_i = Q.x_i$
 endif
 if $f(Q.p_i) < f(Q.p_n)$
  $Q.p_n = Q.p_i$
 endif
endfor
Perform velocity and position updates using equations (2) and (3) for each particle in Q
if $f(\mathbf{b}(1, S_1.p_n)) < f(Q.p_n)$
 for swarm $d \in [1..D]$:
  $S_d.p_n = Q.p_{nd}$
  if $S_d.p_n > x_{dmax} | S_d.p_n < x_{dmin}$
   $S_d.p_n = min(x_{dmax}, max(x_{dmin}, S_d.p_n))$
  endif
 endfor
endif
until stopping condition is met

The dth swarm is denoted as S_d, $S_d.x_i$ denotes the current position of the ith particle in the dth swarm, $S_d.p_n$ and $Q.p_n$ represent the best position in dth swarm and D-dimensional swarm, respectively. Similarly, $S_d.p_i$ and $Q.p_i$ represent the best previous position of particle i in dth swarm and D-dimensional swarm, respectively. The function $b(d, z)$ returns a vector $b(d, z) \equiv (S_1.p_n, S_2.p_n, ..., S_{d-1}.p_n, z, S_{d+1}.p_n, ...S_D.p_n)$ and the context vector P_n or $b(1, S_1.p_n)$ is the concatenation of $S_1.p_n, S_2.p_n, ...S_D.p_n$.

reside. To sum up, there are two advantages about CPSO-H algorithm, fast convergence and obtaining global elements. Here we take an example to illustrate. Fig.1 shows CPSO-H's convergence characteristics and corresponding search vectors, which are repeated three times on the Rotated Schwefel's function. The Rotated Schwefel's function is defined as $f(6)$ in Table I. The other parameters for CPSO-H are: $D = 30$, ω decreases linearly from 0.9 to 0.5, $c_1 = c_2 = 1.49$. There are $D = 30$ swarms in total and each swarm contains 40 particles. The global optimal vector for Rotated Schwefel's function is $X_{opt} = [420.96, 420.96, .., 420.96]$. From Fig.1 we can see that there are three main features about CPSO-H algorithm: first, the three fitness values have a great difference, ranging from 3500 to 5000; second, the fitness values drop very quickly at first 20 iterations but have not very significant improvement after 20 iterations; third, the solution vectors are

diverse, even the worst search in Fig.1 can still obtain some elements on the 17th and 21th dimension which are closer to the global optimal vector.

From Fig.1, we know that CPSO-H algorithm easily gets trapped in local optimum when facing rotated functions which are used to express the nonseparable problems since the interdependency elements on the nonseparable problems can not be changed simultaneously. These interdependency elements falling into different areas depend on different context vectors which are initialized randomly at initialization stage. Therefore, some of the worst search in Fig.1 still have some elements which are closer to the global optimal vector. If the solution vectors are fully utilized, the performance of CPSO-H algorithm can be improved significantly.

B. Motivations for CPSO-EM Algorithm

From above analysis, we know that CPSO-H algorithm converges fast but easily get trapped in local optimum. However, the solution vectors appear diverse, even if the worst search can still find some elements that are closer to the global optimal elements. The other algorithms like [7] and [8] may also obtain useful solution vectors but they converge very slowly, therefore the computational cost is expensive. Since the CPSO-H algorithm converges fast, the first 10 iterations of CPSO-H algorithm is used in order to reduce the computational cost. Moreover, each swarm contains only 10 particles instead of 40 to have a fast convergence and reduce the computational cost further. Fig.2 shows ten times of CPSO-H algorithm in 10 iterations on Rotated Schwefel's function with $D = 30$. It is obvious from Fig.2 that there always exist elements that are closer to the global value 420.96. Imagining all the elements that are closer to the global value 420.96 are chosen as context vector, the performance of CPSO-H algorithm will be improved significantly. This is the reason why we propose the CPSO-EM algorithm where elimination mechanism (EM) memory is introduced to find the best combination of solution vectors by using store, selection and replacement processes.

C. CPSO-EM Algorithm

Since the solution vectors as shown in Fig.2 contain many useful elements that are closer to global ones, EM memory is created to store all these vectors from which useful elements should be extracted. Assuming that there are N solution vectors stored in EM as shown in (4). The method for extracting useful elements from EM is to use iterative algorithm called CPSO-EM. We randomly select elements and form a vector, then this vector is treated as a context vector for CPSO-H algorithm to obtain a new vector X_{new}. If the fitness value of X_{new} is better than that of the worst vector stored in EM, the worst vector should be replaced by X_{new} since a better vector is supposed to have more elements than global ones. Here the vector obtained by randomly selection elements can not be treated as the new vector to compare but as the context vector for CPSO-H algorithm. The reason is due to the fact that the test functions are usually very complex, even if changing one element of the vector may increase the fitness

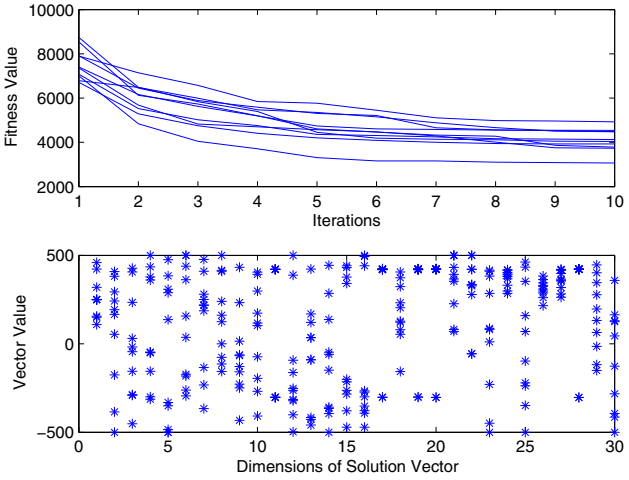

Fig. 2. CPSO-H's convergence characteristics and solution vectors on Rotated Schwefel's function in 10 iterations with $D = 30$. The algorithm repeats ten times.

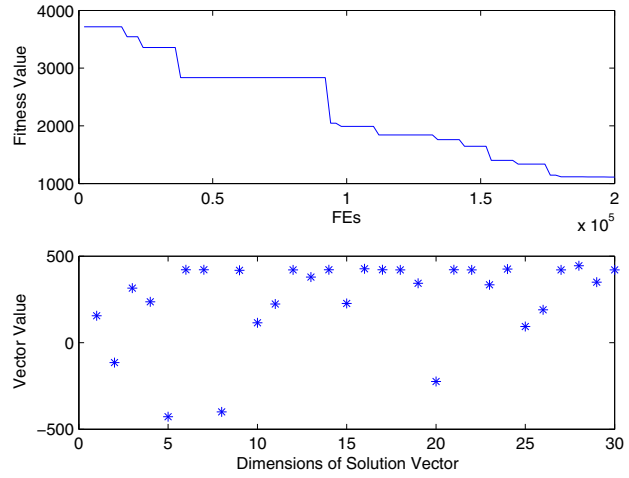

Fig. 3. CPSO-EM's convergence characteristics and solution vector on Rotated Schwefel's function with $D = 30$ and the size of EM equals 10.

value significantly. Imaging a vector contains many elements that are global ones, the fitness value may still be bigger than that of the worst vector on EM and therefore the context vector should be optimized by CPSO-H to obtain new vector X_{new}. The pseudocode of CPSO-EM is shown in Algorithm 2.

$$EM = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} & f(X_1) \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nD} & f(X_n) \\ \vdots & \vdots & & \vdots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} & f(X_N) \end{bmatrix} \quad (4)$$

Since the CPSO-EM algorithm repeats CPSO-H algorithm for EM initialization and extracting useful elements from EM memory which contains all the calculations of FEs, the

Algorithm 2: Pseudocode for the generic CPSO-EM algorithm.

Step 1. Initializing the elimination mechanism (EM) memory
for $n \in [1..N]$:
repeat Algorithm 1, the obtained solution vector X_n is stored in EM memory.
endfor
Step 2. Extracting useful elements from EM memory
repeat:
for each dimension $d \in [1..D]$:
choose any one value from EM memory
endfor
concatenate the values chosen from EM as context vector P_n
replace the context vector of Algorithm 1 as the new vector P_n, then execute Algorithm 1 and obtain a new solution vector X_{new}.
if the fitness value of X_{new} is better than that of the worst in EM replace the worst vector in EM with X_{new}.
endif
until stopping condition is met

computational cost is mainly focused on CPSO-H algorithm and therefore the number of FEs in each CPSO-H algorithm is crucial for the performance of CPSO-EM. The number of FEs implemented on CPSO-H should make the global elements found. From Fig.2 we can see that the global elements can be obtained in 10 iterations. We also tested the other functions and found that operating 10 iterations for each CPSO-H is sufficient due to the reduced search space, because there is only one element optimized each time in CPSO-H. Another parameter that affects the performance of CPSO-EM is EM memory size. A smaller one is inclined to make the CPSO-EM algorithm converge fast and a larger memory size is supposed to make the CPSO-EM algorithm converge slowly but obtain better results. Fig.3 shows the CPSO-EM's convergence characteristics and solution vector on Rotated Schwefel's function with $D = 30$ and the size of EM equals 10. Compared with Fig.2, we can see that the search performance is improved significantly and there are more elements finding the global ones.

IV. Experimental Results and Comparisons

A. Test Functions

In order to test CPSO-EM's performance in different environment, we choose six rotated nonseparable functions tested in 30 and 100 dimensions. The details of these functions are given in Table I. The correlation elements among the Rosenbrock function are in sequence and each element has two correlation elements that are close to it. The correlation elements of the other five rotated functions are created randomly by an orthogonal matrix M. 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 correlation vector, because all elements in vector Y will be affected once one element in vector X changes.

B. Parameter Settings for the Involved PSO Algorithms

In the following part, we briefly describe three algorithms taken from the literature to compare with CPSO-EM. The configuration of these three algorithms is given in Table II.

TABLE II
PSO Algorithms for Comparison

Algorithm	*Parameter Settings*
FIPS [19]	$\chi = 0.7298,\ \sum c_i = 4.1$
CPSO-H$_k$ [6]	$\omega : 0.4 \sim 0.9,\ c_1 = c_2 = 1.494,\ k = 6$
CLPSO [2]	$\omega : 0.2 \sim 0.9,\ Pc_i : 0.05 \sim 0.5$ $c_1 = c_2 = 1.494$

The first FIPS uses all neighbor particles to influence the flying velocity [19]. The second CPSO-H$_k$ uses k subcomponent swarms to search each subcomponent separately and is combined with the original PSO to improve the performance on multimodal problems [6]. The third CLPSO is proposed for solving multimodal problems and its learning strategy is to use all the other particles' best historical information to update one particle's velocity [2].

For the above three algorithms, the initial population is set 40 in 30 and 100 dimensions. The maximal number of FEs is used as the stop criteria which are set 200,000 and 500,000 for 30 and 100 dimensions, respectively. The parameter settings for CPSO-EM are $\omega : 0.4 \sim 0.9, c_1 = c_2 = 1.494$, initial population is 10 for each CPSO-H algorithm in order to reduce the computational cost and the size of EM is set 10.

C. Results for the 30 Dimensional Problems

In order to fairly compare CPSO-EM with other 3 peer algorithms, all algorithms are implemented and run 30 times on the six test functions. Table III shows the means and variances of the four algorithms on the six test functions with dimensions $D = 30$. The best results for test functions are shown in bold. Fig.4 shows the median convergence characteristics of the six test functions in 30 dimensions. From Table III and Fig.4, we can see that CPSO-H$_k$ performs the worst on nonseparable problems but converges very fast. FIPS is a local version of PSO which is very suitable for solving multimodal problems. It performs well on f_2 but poorly on f_3 and f_4 and therefore the performance of FIPS is highly dependent on different test functions. CLPSO works well on all these six functions and obtains the best on f_1 and f_2 but converges slowly. The CPSO-EM outperforms the other algorithms on the six functions except f_1 and f_2 where CLPSO works well. Moreover, it is obvious from Fig.4 that CPSO-EM can obtain the best values among the whole searching process.

D. Results for the 100 Dimensional Problems

The experiments implemented in 30 dimensional problems are repeated in 100 dimensional problems. The experimental results and the convergence characteristics are shown in Table IV and Fig.5, respectively. From Table IV we can see that FIPS may not be suitable for large dimensional problems. the performance is deteriorated a lot compared with that of 30 dimensional problems for reference. The performance of CPSO-H$_k$ and CLPSO in 100 dimensional problems are very similar to that in 30 dimensional problems and CPSO-EM still surpasses the others on functions f_3 to f_6. Moreover,

TABLE I
TEST FUNCTIONS, WHERE f_{min} IS THE MINIMUM VALUE

Name	*Test Function*	*Search Range*	f_{min}
Rosenbrock	$f_1(x) = \sum_{i=1}^{D-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$[-2.048, 2.048]^D$	0
Rotated Ackley	$f_2(x) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi y_i)\right) + 20 + e$ Where: $Y = M * X$, M is an orthogonal matrix	$[-32.768, 32.768]^D$	0
Rotated Rastrigin	$f_3(x) = \sum_{i=1}^{D} y_i^2 - 10cos(2\pi y_i) + 10$ Where: $Y = M * X$, M is an orthogonal matrix	$[-5.12, 5.12]^D$	0
Rotated Noncontinuous Rastrigin	$f_4(x) = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10)$ Where: z_i is expressed as: $z_i = \begin{cases} y_i & \lvert y_i\rvert < \frac{1}{2} \\ \frac{round(2y_i)}{2} & \lvert y_i\rvert >= \frac{1}{2} \end{cases} for\ i = 1, 2, ..., D.$ $Y = M * X$, M is an orthogonal matrix	$[-5.12, 5.12]^D$	0
Rotated Weierstrass	$f_5(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{kmax}[a^k cos(2\pi b^k (y_i + 0.5))]\right) - D\sum_{k=0}^{kmax}[a^k cos(\pi b^k)]$ Where: $Y = M * X$, $a = 0.5$, $b = 3$, $kmax = 20$, M is an orthogonal matrix	$[-0.5, 0.5]^D$	0
Rotated Schwefel	$f_6(x) = 418.9829 \times D - \sum_{i=1}^{D} z_i$ Where: z_i is expressed as: $z_i = \begin{cases} y_i sin\left(\lvert y_i\rvert^{\frac{1}{2}}\right) & if\ \lvert y_i\rvert <= 500 \\ 0 & otherwise \end{cases} for\ i = 1, 2, ..., D.$ $y = y' + 420.96$, $Y' = M * (X - 420.96)$. M is an orthogonal matrix	$[-500, 500]^D$	0

TABLE III
RESULTS FOR THE 30 DIMENSIONAL PROBLEMS

Functions	*FIPS*	*CPSO-H_k*	*CLPSO*	*CPSO-EM*
f_1	$2.43e{+}001 \pm 7.16{+}000$	$2.17e{+}001 \pm 8.13e{+}000$	$\mathbf{1.71e{+}001 \pm 1.69e{+}001}$	$1.83e{+}001 \pm 1.06e{+}001$
f_2	$2.98e{-}004 \pm 2.56e{-}004$	$1.62e{+}000 \pm 1.76{+}000$	$\mathbf{2.42e{-}005 \pm 3.19{-}005}$	$1.73e{-}002 \pm 1.03{-}002$
f_3	$1.12e{+}002 \pm 3.02e{+}001$	$8.35e{+}001 \pm 6.71e{+}001$	$3.42e{+}001 \pm 6.19e{+}000$	$\mathbf{2.81e{+}001 \pm 1.11e{+}001}$
f_4	$1.26e{+}002 \pm 4.25e{+}001$	$7.36e{+}001 \pm 3.56e{+}001$	$3.82e{+}001 \pm 1.05e{+}001$	$\mathbf{2.64e{+}001 \pm 9.21e{+}000}$
f_5	$2.03e{+}001 \pm 5.49{+}000$	$1.12e{+}001 \pm 5.81{+}000$	$4.26e{+}000 \pm 1.46{+}000$	$\mathbf{3.25e{+}000 \pm 2.25e{+}000}$
f_6	$2.89e{+}003 \pm 1.39{+}003$	$3.73e{+}003 \pm 1.73{+}003$	$2.24e{+}003 \pm 7.93{+}002$	$\mathbf{1.45e{+}003 \pm 8.23{+}002}$

TABLE IV
RESULTS FOR THE 100 DIMENSIONAL PROBLEMS

Functions	*FIPS*	*CPSO-H_k*	*CLPSO*	*CPSO-EM*
f_1	$9.78e{+}001 \pm 2.31e{+}001$	$1.22e{+}002 \pm 3.81e{+}001$	$\mathbf{9.15e{+}001 \pm 4.72e{+}000}$	$9.32e{+}001 \pm 1.17e{+}001$
f_2	$5.58e{-}002 \pm 2.12e{-}002$	$2.14e{+}000 \pm 1.86{+}000$	$\mathbf{4.43e{-}005 \pm 2.16{-}005}$	$2.15e{-}002 \pm 1.77{-}002$
f_3	$8.77e{+}002 \pm 1.82e{+}002$	$3.26e{+}002 \pm 1.32e{+}002$	$2.13e{+}002 \pm 7.26e{+}001$	$\mathbf{1.62e{+}002 \pm 4.21e{+}001}$
f_4	$8.12e{+}002 \pm 1.57e{+}002$	$2.63e{+}002 \pm 6.53e{+}001$	$2.26e{+}002 \pm 3.35e{+}001$	$\mathbf{1.39e{+}002 \pm 2.19e{+}001}$
f_5	$1.32e{+}002 \pm 2.13{+}001$	$3.83e{+}001 \pm 7.22{+}000$	$3.36e{+}001 \pm 4.10{+}000$	$\mathbf{1.02e{+}001 \pm 3.16e{+}000}$
f_6	$2.62e{+}004 \pm 7.18{+}003$	$1.26e{+}004 \pm 5.03{+}003$	$1.06e{+}004 \pm 3.16{+}003$	$\mathbf{5.86e{+}003 \pm 3.17{+}003}$

with increased dimensions, the CPSO-EM has more prominent advantages in terms of convergence speed and the best fitness value.

V. CONCLUSION

This paper presents a novel algorithm CPSO-EM where the CPSO-H algorithm and EM memory are incorporated together. In order to fully utilize the two advantages of CPSO-H: fast convergence and obtaining global elements of solution vector, EM memory is introduced from which the useful elements are extracted. The CPSO-EM reduces the search space by adopting the CPSO-H algorithm, this advantage is more prominent with higher dimensions. Meanwhile, since EM memory increases the diversity of particles, CPSO-EM can break out of local optimum and therefore it is very suitable for solving nonseparable problems. From comparisons of the four algorithms on six nonseparable test functions, CPSO-EM has shown the strong ability in solving nonseparable multimodal problems especially with higher dimensions.

REFERENCES

[1] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.,* vol. 1, no. 1, pp. 33-58, 2007.

[2] J. J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar. "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.,* vol. 10, no. 3, pp. 281-295, Jun. 2006.

[3] T. M. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.,* vol. 10, no. 4, pp. 459-472, Aug. 2006.

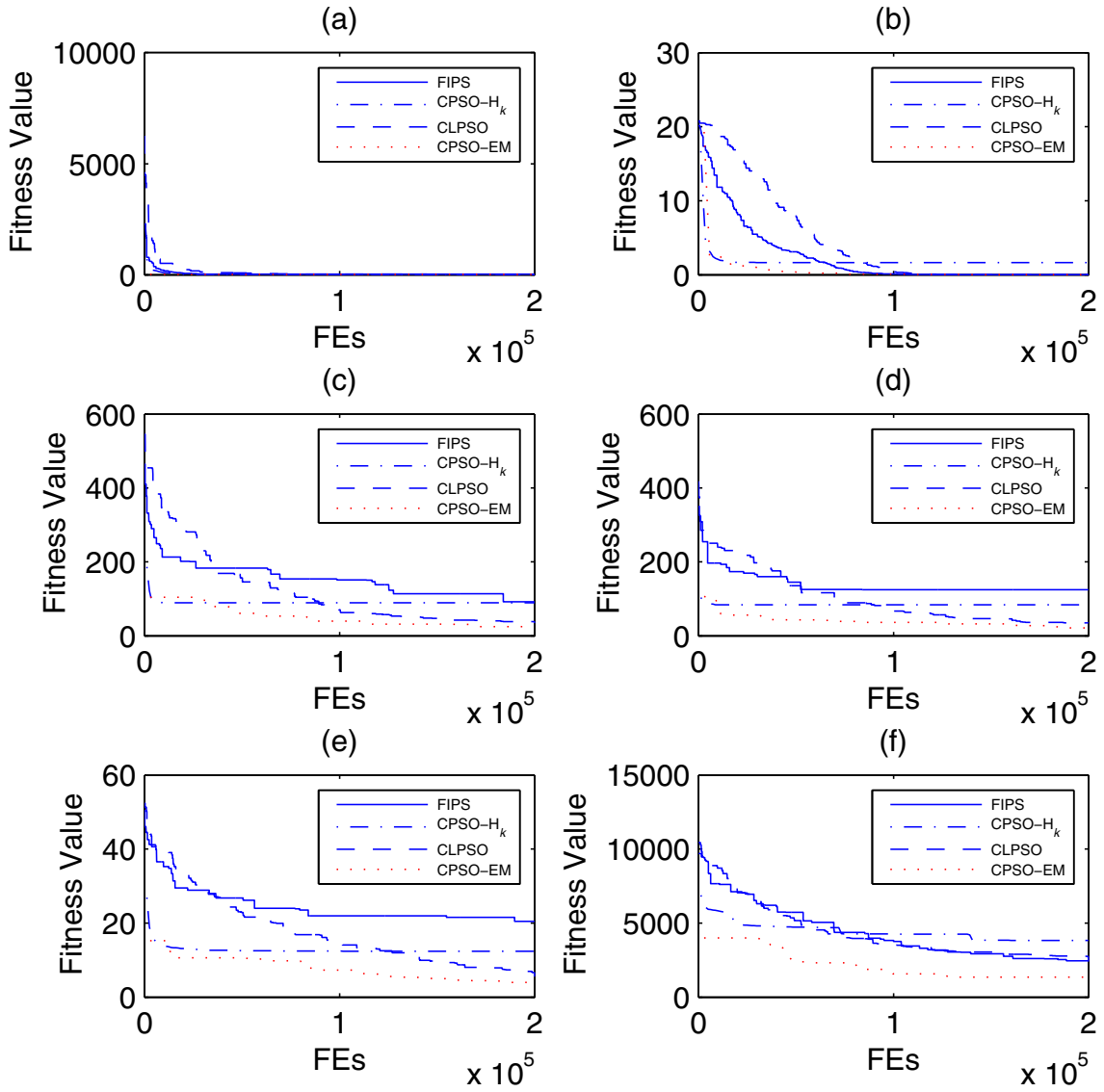

Fig. 4. The median convergence characteristics of 30 dimensional test functions. (a) Rosenbrock's function. (b) Rotated Ackley's function. (c) Rotated Rastrigin's function. (d) Rotated Noncontinuous Rastrigin's function. (e) Rotated Weierstrass's function. (f) Rotated Schwefel's function.

[4] X. Chen and Y.M. Li "A modified PSO structure resulting in high exploration ability with convergence guaranteed," *IEEE Trans. Syst., Man, Cybern. B,* vol. 37, no. 5, pp. 1271-1289, Oct. 2007.

[5] T. M. Blackwell and P. Bentley, "Don't push me! collision-avoiding swarms," in *Proc. Congr. Evol. Comput.,* 2002, vol.2, pp. 1691-1696.

[6] F. van den Bergh and A.P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.,* vol. 8, no. 3, pp. 225-239, Jun. 2004.

[7] X. Li, and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *Proc. IEEE Congr. Evol. Comput.,* May 2009, pp. 1546-1553.

[8] X. Li, and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.,* vol. 16, no. 2, pp. 210-224, Apr. 2004.

[9] R.C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th. Int. Symp. Micromachine Human Sci.,* Nagoya, Japan, 1995, pp. 39-43.

[10] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks,* 1995, pp. 1942-1948.

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.,* 1998, pp. 69-73.

[12] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimizer", in *Proc. 7th Conf. Evol. Programming,* New York, 1998, pp.591-600.

[13] Z.H. Zhan, J. Zhang, Y. Li, and H.S. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B,* vol 39, no. 6, pp. 1362-1381, Dec. 2009.

[14] P.N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proc. Congr. Evol. Comput.,* July 1999, pp. 1958-1962

[15] C. Li, S.X. Yang and T.T. Nguyen, "A Self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Syst., Man, Cybern. B,* vol. 42, no. 3, pp. 627-643, Jun. 2012.

[16] Z. H. Zhan, J. Zhang, Y. Li, and H. S. Chung, "Adaptive particle swarm

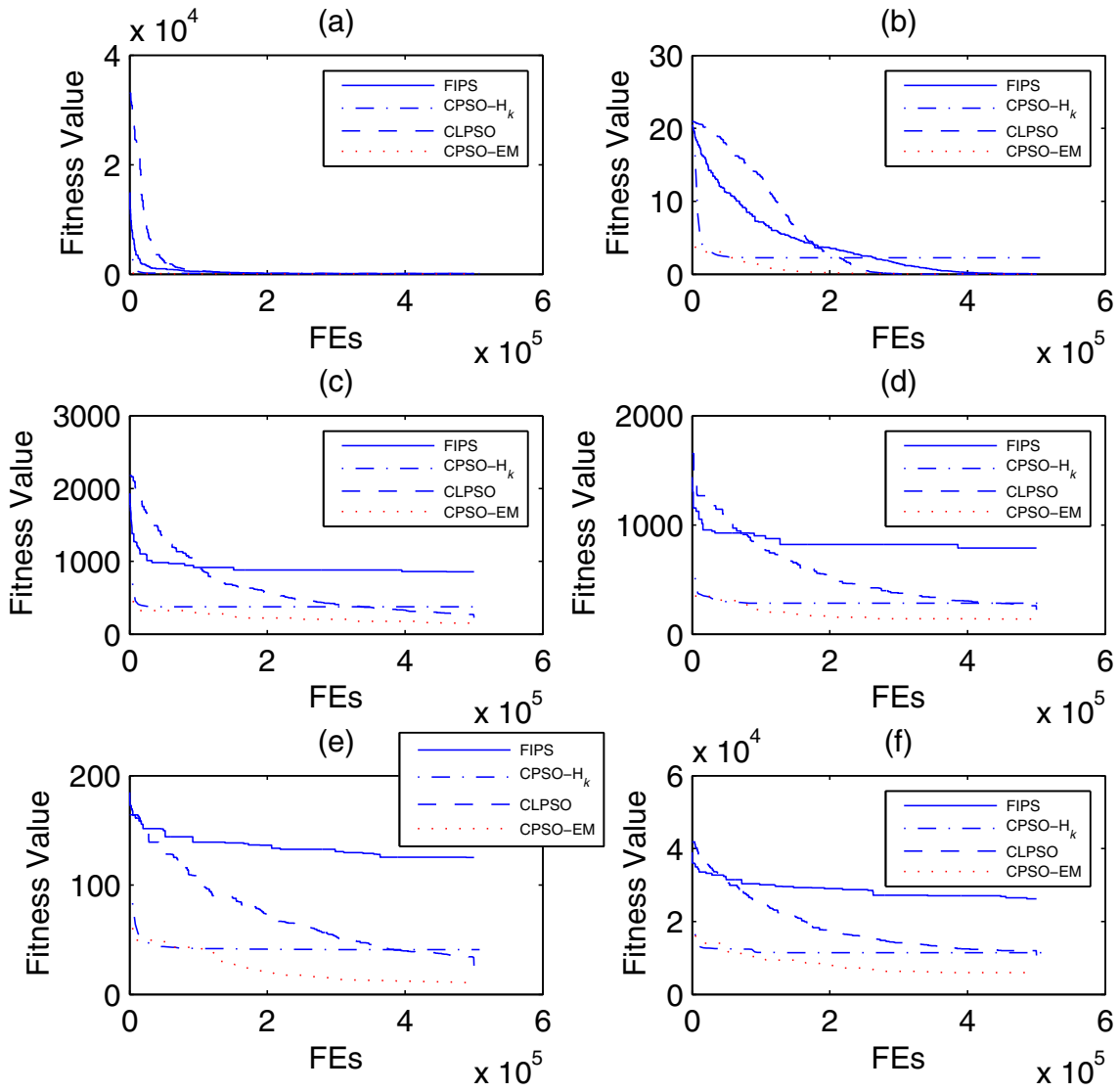

Fig. 5. The median convergence characteristics of 100 dimensional test functions. (a) Rosenbrock's function. (b) Rotated Ackley's function. (c) Rotated Rastrigin's function. (d) Rotated Noncontinuous Rastrigin's function. (e) Rotated Weierstrass's function. (f) Rotated Schwefel's function.

optimization," *IEEE Trans. Syst., Man, Cybern. B,* vol 39, no. 6, pp. 1362-1381, Dec. 2009.

[17] S. Hsieh, T. Sun, C. Liu and S. Tsai, "Efficient population utilization strategy for particle swarm optimizer," *IEEE Trans. Syst., Man, Cybern. B,* vol. 39, no. 2, pp. 444-456, Apr. 2009.

[18] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proc. Congr. Evol. Comput.,* 1999, pp. 1958-1962.

[19] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.,* vol. 8, no.3, pp.204-210, Jun. 2004.

[20] S.H. Ling, H.H.C. Iu, K.Y. Chan, H.K.Lam, B.C.W. Yeung, and F. H. Leung "Hybrid particle swarm optimization with wavelet mutation and its industrial applications," *IEEE Trans. Syst., Man, Cybern. B,* vol. 38, no. 3, pp. 743-763, Jun. 2008.

[21] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proc. Swarm Intell. Symp.,* 2003, pp. 174-181.

[22] P.J. Angeline,"Using selection to improve particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.,* 1998, pp. 84-89.

[23] C.F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Trans. Syst., Man, Cybern. B,* vol. 34, no. 2, pp. 997-1006, 2004.

[24] S.Z. Zhao, P.N. Suganthan, and S. Das, "Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search," in *Proc. IEEE Congr. Evol. Comput.,* 2010, pp. 1-8.

[25] G. Zhang and Y.M. Li, "Orthogonal experimental design method used in particle swarm optimization for multimodal problems," in *The Sixth Int. Conf. Adv. Comput. Intelli.,* October 19-21, 2013, pp.183-188.