Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 215472, 16 pages
http://dx.doi.org/10.1155/2014/215472

Research Article

An Adaptive Hybrid Algorithm Based on Particle Swarm Optimization and Differential Evolution for Global Optimization

Xiaobing Yu,[1,2] Jie Cao,[1,2] Haiyan Shan,[1] Li Zhu,[1] and Jun Guo[3]

[1] *China Institute of Manufacturing Development, Nanjing University of Information Science & Technology, Nanjing 210044, China*
[2] *Collaborative Innovation Center on Forecast and Evaluation of Meteorological Disasters, Nanjing University of Information Science & Technology, Nanjing 210044, China*
[3] *School of Mechanic and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China*

Correspondence should be addressed to Xiaobing Yu; yuxb111@163.com

Received 30 September 2013; Accepted 17 November 2013; Published 9 February 2014

Academic Editors: T. Chen, Q. Cheng, and J. Yang

Particle swarm optimization (PSO) and differential evolution (DE) are both efficient and powerful population-based stochastic search techniques for solving optimization problems, which have been widely applied in many scientific and engineering fields. Unfortunately, both of them can easily fly into local optima and lack the ability of jumping out of local optima. A novel adaptive hybrid algorithm based on PSO and DE (HPSO-DE) is formulated by developing a balanced parameter between PSO and DE. Adaptive mutation is carried out on current population when the population clusters around local optima. The HPSO-DE enjoys the advantages of PSO and DE and maintains diversity of the population. Compared with PSO, DE, and their variants, the performance of HPSO-DE is competitive. The balanced parameter sensitivity is discussed in detail.

1. Introduction

Evolutionary algorithms (EAs), inspired by the natural evolution of species, have been successfully applied to solve numerous optimization problems in diverse fields [1]. Particle swarm optimization (PSO) and differential evolution (DE) are two stochastic, population-based optimization EAs [2].

PSO was introduced by Kennedy and Eberhart in 1995 [3, 4]. PSO uses a simple mechanism that mimics swarm behavior in birds flocking and fish schooling to guide the particles to search for globally optimal solutions. As PSO is easy to implement, it has rapidly progressed in recent years with many successful applications in solving real-world optimization problems. During the last decade, PSO algorithm has been paid much attention to in various areas [5–19] due to its effectiveness in handling optimization problems. Unfortunately, the PSO algorithm suffers from the premature convergence problem which does exist in complex optimization issues. In [17–19], some methods for tuning parameters including inertia weights and acceleration coefficients for PSO have been proposed to enhance PSO's search performance. A comprehensive learning PSO (CLPSO) algorithm was proposed in [14], which shows its superiority in dealing with multimodal functions.

DE is a simple yet powerful EA for global optimization introduced by Storn and Price [20]. The DE algorithm has gradually become more popular and has been used in many practical cases, mainly because it has demonstrated good convergence properties and is principally easy to understand [21]. DE has been successfully applied in diverse fields of engineering [22–26]. The performance of the conventional DE algorithm highly depends on the chosen trial vector generation strategy and associated parameter values used. Inappropriate choice of strategies and parameters may lead to premature convergence, which have been extensively demonstrated in [27]. In the past decade, DE researchers have

suggested many empirical guidelines for choosing trial vector generation strategies and their associated control parameter settings [1, 28–31].

Although DE and PSO have been successfully applied to a wide range of problems including test and real life problems, both have certain shortcomings associated with them which sometimes deteriorate the performance of algorithms. The major problem is the lack of diversity resulting in a suboptimal solution or a slow convergence rate [32]. In order to improve the performance of these algorithms, one of the class of modified algorithms consists of the hybridization of algorithms, where the two algorithms are combined together to form a new algorithm. DE is applied to each particle for a finite number of iterations to determine the best particle which is then included into the population [33]. DE is applied to the best particle obtained by PSO [34]. A hybrid version of PSO and DE is proposed which is named Barebones DE [35]. The evolution candidate solution is generated either by DE or by PSO according to some fixed probability distribution [36]. A hybrid metaheuristic is designed so as to preserve the strengths of both algorithms [32].

However, it is worth mentioning that, in almost all the hybrid works mentioned above, the convergence rate is not fast enough or the global search performance is not satisfactory. To achieve the goal, a novel adaptive hybrid algorithm based on PSO and DE (HPSO-DE) is formulated by developing a balanced parameter between PSO and DE. Adaptive mutation is carried out to current population when the population clusters around local optima. The HPSO-DE enjoys the advantages of PSO and DE and maintains diversity of the population. With the efforts, the HPSO-DE has the ability to jump out of the local optima.

2. PSO and DE

2.1. PSO. In the standard PSO, a swarm consists of m individuals (called particles) that fly around in an n-dimensional search space. The position of the ith particle at the tth iteration is used to evaluate the particle and represent the candidate solution for the optimization problem. It can be represented as $X_i^t = [x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t]$, where x_{ij}^t is the position value of the ith particle with respect to the jth dimension ($j = 1, 2, \ldots, n$). During the search process, the position of a particle is guided by two factors: the best position visited by itself (p_{best}) denoted by $P_i^t = [p_{i1}^t, p_{i2}^t, \ldots, p_{in}^t]$ and the position of the best particle found so far in the swarm (g_{best}) denoted by $G^t = [g_1^t, g_2^t, \ldots, g_n^t]$. The new velocity (denoted by $V_i^t = [v_{i1}^t, v_{i2}^t, \ldots, v_{in}^t]$) and position of particle i at the next iteration are calculated according to

$$v_{ij}^{t+1} = w \times v_{ij}^t + c_1 \times r_1 \times \left(p_{ij}^t - x_{ij}^t\right) + c_2 \times r_2 \times \left(g_j^t - x_{ij}^t\right), \tag{1}$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1}, \tag{2}$$

where w is the inertia weight, c_1, and c_2 are, respectively, the cognitive and social learning parameters, and r_1, and r_2 are random numbers between $(0, 1)$. Based on the above equations, the particle can fly through search space toward p_{best} and g_{best} in a navigated way [16, 17].

2.2. PSO Variants

2.2.1. PSO-w. In the PSO algorithm, proper control of global exploration and local exploitation is an important issue. In general, the higher values of inertia weight w help in exploring the search space more thoroughly in the process and benefit the global search, while lower values help in the local search around the current search space. The major concern of this linear PSO is to avoid the premature convergence in the early period of the search and to enhance convergence to the global optimum solution during the latter period of the search. The concept of linearly decreasing inertia weight was introduced in [17] and is given by

$$w = w_{\max} - \left(w_{\max} - w_{\min}\right)\left(\frac{\text{iter}}{\text{iter}_{\max}}\right), \tag{3}$$

where iter is the current iteration number and $\text{iter}_{\max}$ is the maximum number of iteration. Usually the value of w is between 0.9 and 0.4. Therefore, the particle is to use lager inertia weight during the initial exploration and gradually reduce its value as the search proceeds in further iterations. According to the research [37], the inertia weight is adjusted by (4). The nonlinear descending can achieve faster convergence speed than that with linear inertia weight:

$$w = \left(w_1 - w_2\right) \times \frac{\left(\text{iter} - \text{iter}_{\max}\right)^2}{\left(\text{iter}_{\max}\right)^2} + w_2, \tag{4}$$

where w_1 and w_2 are the initial and final inertia weight.

2.2.2. PSO-TVAC. Although linear PSO can locate satisfactory solution at a markedly fast speed, its ability to fine-tune the optimum solution is limited, mainly due to the lack of diversity at the latter stage of evolution process. In population-based optimization methods, the guideline is to encourage the individuals to roam through the entire search space during the early period of the search, without clustering around local optima. During the later period, convergence towards the global optima is encouraged [3]. With this view, a novel strategy in which time-varying acceleration coefficients are employed by changing the acceleration coefficients with time is proposed [16, 38]. With a large cognitive component and small social component at the beginning, particles are allowed to move around the search space, instead of moving toward the population best. On the other hand, a small cognitive component and a large social component allow the particles to converge to the global optima in the latter part of the optimization. This approach is referred to as PSO-TVAC. This modification can be mathematically represented as follows:

$$\begin{aligned} c_1 &= \left(c_{1f} - c_{1i}\right)\left(\frac{\text{iter}}{\text{iter}_{\max}}\right) + c_{1i}, \\ c_2 &= \left(c_{2f} - c_{2i}\right)\left(\frac{\text{iter}}{\text{iter}_{\max}}\right) + c_{2i}, \end{aligned} \tag{5}$$

where c_{1i}, c_{1f}, c_{2i}, and c_{2f} are initial and final values of cognitive and social acceleration factors, respectively; usually $c_{1i} = c_{2f} = 2.5$ and $c_{1f} = c_{2i} = 0.5$ [38].

2.3. DE. DE is proposed by Storn and Price [20]. It is an effective, robust, and simple global optimization algorithm. According to frequently reported experimental studies, DE has shown better performance than many other EAs in terms of convergence speed and robustness over several benchmark functions and real-world problems [39].

In DE, there are three operators: mutation, crossover, and selection. Initially, a population is generated randomly with uniform distribution; then the mutation, crossover, and selection operators are applied to generate a new population. Trial vector generation is a crucial step in DE process. The two operators mutation and crossover are used to generate the trial vectors. The selection operator is used to select the best trial vector for the next generation. The initialization and DE operators are explained briefly as follows [40].

DE starts with a population of NP D-dimensional candidate solutions which may be represented as $X_{i,G}(i = 1, 2, \ldots, \mathrm{NP}) = \{x^1_{i,G}, x^2_{i,G}, \ldots, x^D_{i,G}\}$, where index i denotes the ith individual of the population, G denotes the generation to which the population belongs, and D is the dimension of the population.

The initial population should try to cover the entire search space as much as possible by uniformly randomizing individuals within the search space constrained by the minimum $X_{\min} = \{x^1_{\min}, x^2_{\min}, \ldots, x^D_{\min}\}$ and maximum $X_{\max} = \{x^1_{\max}, x^2_{\max}, \ldots, x^D_{\max}\}$ bounds. Thus, the initial population can be described as follows:

$$x_{i,0} = x_{\min} + \mathrm{rand}(0,1) \times (x_{\max} - x_{\min}), \tag{6}$$

where $\mathrm{rand}(0,1) \in [0,1]$ is a uniformly distributed random variable [40, 44].

(1) Mutation. After initialization, DE utilizes the mutation operation to generate a trial vector $V_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ with respect to each individual in the current population. $V_{i,G}$ can be produced by certain mutation strategy. For example, the five most frequently mutation strategies implemented in the DE are listed as follows [1]:

DE/rand/1:

$$V_{i,G} = X_{r^i_1,G} + F \cdot \left(X_{r^i_2,G} - X_{r^i_3,G}\right), \tag{7}$$

DE/best/1:

$$V_{i,G} = X_{\mathrm{best},G} + F \cdot \left(X_{r^i_1,G} - X_{r^i_2,G}\right), \tag{8}$$

DE/rand-to-best/1:

$$V_{i,G} = X_{i,G} + F \cdot \left(X_{\mathrm{best},G} - X_{i,G}\right) + F \cdot \left(X_{r^i_1,G} - X_{r^i_2,G}\right), \tag{9}$$

DE/best/2:

$$V_{i,G} = X_{\mathrm{best},G} + F \cdot \left(X_{r^i_1,G} - X_{r^i_2,G}\right) + F \cdot \left(X_{r^i_3,G} - X_{r^i_4,G}\right), \tag{10}$$

DE/rand/2:

$$V_{i,G} = X_{r^i_1,G} + F \cdot \left(X_{r^i_2,G} - X_{r^i_3,G}\right) + F \cdot \left(X_{r^i_4,G} - X_{r^i_5,G}\right). \tag{11}$$

The indices r^i_1, r^i_2, r^i_3, r^i_4, and r^i_5 are mutually exclusive integers randomly generated within the range $[0, 1]$, which are also different from the index i [1]. F is the mutation scale factor which is used in controlling the amplification of the differential variation [40].

(2) Crossover. The crossover operation is introduced to increase the diversity of the target vectors. After the mutation phase, the crossover operation is applied to $V_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ and $X_{i,G} = \{X^1_{i,G}, X^2_{i,G}, \ldots, X^D_{i,G}\}$ to generate a trial vector $U_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \ldots, u^D_{i,G}\}$ as follows:

$$u^j_{i,G} = \begin{cases} v^j_{i,G} & \text{if } \mathrm{rand}_j[0,1) \le \mathrm{CR} \text{ or } \left(j = j_{\mathrm{rand}}\right), \\ x^j_{i,G} & \text{others.} \end{cases} \tag{12}$$

$\mathrm{CR} \in [0,1]$ is the crossover constant, which has to be determined by the user. $j_{\mathrm{rand}} \in [1, D]$ is a randomly chosen index which ensures that the trial vector $U_{i,G}$ will differ from $X_{i,G}$ by at least one parameter.

(3) Selection. If the generated trial vector $u^j_{i,G}$ exceeds the corresponding upper and lower bounds, we randomly and uniformly reinitialize it within the search range. Then the fitness values of all trial vectors are evaluated [45].

After that, a greedy selection scheme is employed:

$$x_{i,G+1} = \begin{cases} u_{i,G} & \text{if } f(u_{i,G}) \le f(x_{i,G}), \\ x_{i,G} & \text{otherwise.} \end{cases} \tag{13}$$

If the trial vector $u_{i,G}$ yields a better cost function value than $x_{i,G}$, the $u_{i,G}$ will replace $x_{i,G}$ and enter the population of the next generation; otherwise, the old value $x_{i,G}$ is retained.

2.4. DE Variants. In order to improve the performance of DE, some adaptive DE variants are proposed. jDE was proposed based on the self-adaptation of the scale factor F and the crossover rate CR [29]. The jDE is represented by a D-dimensional vector $X_{i,G}(i = 1, 2, \ldots, \mathrm{NP})$. New control parameters or factors $F_{i,G+1}$ and $\mathrm{CR}_{i,G+1}$ are calculated as

$$F_{i,G+1} = \begin{cases} F_l + \mathrm{rand}_1 \times F_u & \text{if } \mathrm{rand}_2 < \tau_1, \\ F_{i,G} & \text{otherwise,} \end{cases}$$
$$\mathrm{CR}_{i,G+1} = \begin{cases} \mathrm{rand}_3 & \text{if } \mathrm{rand}_4 < \tau_2, \\ \mathrm{CR}_{i,G} & \text{otherwise,} \end{cases} \tag{14}$$

and they produce factors F and CR in a new parent vector. rand_j $(j \in \{1, 2, 3, 4\})$ are uniform random values within the range $[0, 1]$. τ_1 and τ_2 represent probabilities to adjust factors F and CR, respectively. Generally, $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$, $F_u = 0.9$. SaDE gave the first attempt to simultaneously adopt more than one mutation scheme in DE [1]. The main propose is to reduce the problem-solving risk by distributing available

computational resources to multiple search techniques with different biases. SaNSDE can be regarded as an improved version of the SaDE. Its mutation is executed in the same way as SaDE except that only two mutation schemes are used, and the scale factor F in the adopted mutation schemes is generated according to either a Gaussian distribution or a Cauchy distribution [30]. JADE is another recent DE variant, in which a new mutation scheme named "/DE/current-to-pbest" is adopted [31].

3. Proposed HPSO-DE

Similar to other EAs, both of PSO and DE are the population-based iterative algorithms. The PSO and DE can easily get trapped in the local optima when solving complex multimodal problems. These weaknesses have restricted wider applications of them. Therefore, avoiding the local optima has become one of the most important and appealing goals in algorithms research. To achieve the goal, an adaptive hybrid algorithm based on PSO and DE (HPSO-DE) for global optimization is proposed. The algorithm focuses on the convergence of population. When a particle discovers a current optima position, the other particles will draw together to the particle. If the position is the local optima, the algorithm will be convergence and clustered in local optima. The premature may appear. Suppose that the population size of HPSO-DE is NP, the fitness value of ith particle is f_i, and the average fitness value is f_{avg}. The convergence degree is defined as follows:

$$d = \sqrt{\sum_{i=1}^{NP}\left(\frac{f_i - f_{\text{avg}}}{\max\{1, \max_{1\le i\le N}(f_i - f_{\text{avg}})\}}\right)^2}. \quad (15)$$

The parameter d reflects the convergence degree. When the parameter d is large, the algorithm is in random search. On the other hand, the algorithm will get into local optima and the premature maybe occur. In order to evaluate the parameter d, d_c is given as follows, where p is the mutation probability.

$$p = \begin{cases} k, & d < d_c, \\ 0, & \text{others}. \end{cases} \quad (16)$$

Generally, $d_c \in [0.5, 2]$. If the parameter d is less than d_c, the mutation probability p is equal to k. The mutation including PSO and DE is as follows, respectively:

$$g_{\text{best}} = (1 + 0.5\eta)\, g_{\text{best}}, \quad (17)$$

$$X_{i,G} = (1 + 0.5\eta)\, X_{i,G}, \quad (18)$$

where the parameter η obeys Gauss $(0, 1)$ distribution.

The strategy makes HPSO-DE enjoy the advantages of two algorithms and maintain diversity of the population. With the efforts, the HPSO-DE has the ability of jumping out of the local optima. The main procedure of HPSO-DE is presented in Algorithm 1.

One of the most important parameters for the proposed HPSO-DE is the balanced parameter p. In the following, we will make an integrated analysis of the key parameter by comparing the performance of the HPSO-DE in the optimization of several representative functions.

4. Numerical Experiments and Results

4.1. Test Functions. 16 benchmark functions are used to test the performance of HPSO-DE to assure a fair comparison. If the number of test problems is smaller, it will be very difficult to make a general conclusion. Using a test set which is too small also has the potential risk that the algorithm is biased (optimized) toward the chosen set of problems. Such bias might not be useful for other problems of interest. The benchmark functions are given in Table 1. It denotes the ranges of the variables and the value of the global optimum. Functions $f_1 - f_{16}$ are high-dimensional problems. Functions $f_1 - f_6$ are unimodal. Function f_7 is a noisy quadratic function. Functions $f_8 - f_{16}$ are multimodal functions where the number of local minima increases exponentially with the problem dimension [29]. $f_{15} - f_{18}$ are rotated functions. An orthogonal matrix M is generated to rotate a function. The original variable x is left-multiplied by the orthogonal matrix M to get the new rotated variable $y = M \times x$. This variable y is used to compute the fitness value f. When one dimension in x is changed, all dimensions in y will be influenced:

$$f_1 = \sum_{i=1}^{D} x_i^2,$$

$$f_2 = \sum_{i=1}^{D} i x_i^2,$$

$$f_3 = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|,$$

$$f_4 = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2,$$

$$f_5 = \max_i \{|x_i|, 1 \le x_i \le D\},$$

$$f_6 = \sum_{i=1}^{D-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right],$$

$$f_7 = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1),$$

$$f_8 = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right],$$

$$f_9(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos(2\pi y_i) + 10\right),$$

$$y_i = \begin{cases} x_i, & |x_i| < 0.5, \\ \dfrac{\text{round}(2x_i)}{2}, & |x_i| \ge 0.5, \end{cases}$$

Step 1. Initialize parameters: *PSO-Flag=false*; *DE-Flag=false*; the mutation probability of PSO: PSO_p; the mutation probability of DE: DE_p; the balanced parameter between PSO and DE: p. Convergence evaluation parameter d_c.

Step 2. generation counter $G = 0$ and randomly initialize a population of NP individuals $P_G = \{X_{1,G}, ..., X_{\mathrm{NP},G}\}$ with $X_{i,G} = \{x_{i,G}^1, ..., x_{i,G}^D\}$, $i = 1, ..., \mathrm{NP}$ uniformly distributed in the range $[X_{\min}, X_{\max}]$, where $X_{\min} = \{x_{\min}^1, x_{\min}^2, ..., x_{\min}^D\}$ and $X_{\max} = \{x_{\max}^1, x_{\max}^2, ..., x_{\max}^D\}$.

Step 3. Evaluation the population and Identify the best position.

Step 4. While stopping criterion is not satisfied

Generate Random number rand in (0, 1).

If rand < p

Set *DE-Flag=true*; %%Active jDE for current population

for $i = 1$ to the NP do

Update F and R according to (14);

Generate $V_{i,G}$ using (7)

Generate the trial vector $U_{i,G}$ by (12)

End for

Else

Set *PSO-Flag=true*; %%Active PSO for current population

Update w using (4)

for $i = 1$ to the NP do

Update particle Velocity and Position according to (1), (2) respectively.

Set the values of position to the trial vector $U_{i,G}$.

End for

End if

Step 4.1. Randomly reinitialize the trial vector $U_{i,G}$ within the search space if any variable is outside its boundaries.

Step 4.2. Selection

for $i = 1$ to NP

Evaluate the trial vector $U_{i,G}$

If $f(U_{i,G}) \le f(X_{i,G})$

$X_{i,G+1} = U_{i,G}, f(X_{i,G+1}) = f(U_{i,G})$

$P_{i,p\mathrm{best}} = U_{i,G}, f(P_{i,p\mathrm{best}}) = f(U_{i,G})$

If $f(U_{i,G}) < f(X_{\mathrm{best},G})$

$X_{\mathrm{best},G} = U_{i,G}, f(X_{\mathrm{best},G}) = f(U_{i,G})$

$p_g = U_{i,G}, f(p_g) = f(U_{i,G})$

End if

End if

End for

Step 4.3. Calculate parameter d using (15)

If parameter meets the requirement of (16)

Generate a random number rand in (0, 1).

If rand is less than PSO_p and *PSO-Flag* is true

Update g_{best} using (17)

End if

If rand is less than DE_p and *DE-Flag* is true

Update $X_{i,G+1}$ using (18)

End if

End if

Step 4.4. Increment the generation count $G = G + 1$;

Reset *PSO-Flag=false*; *DE-Flag=false;*

Step 5. End while

Algorithm 1

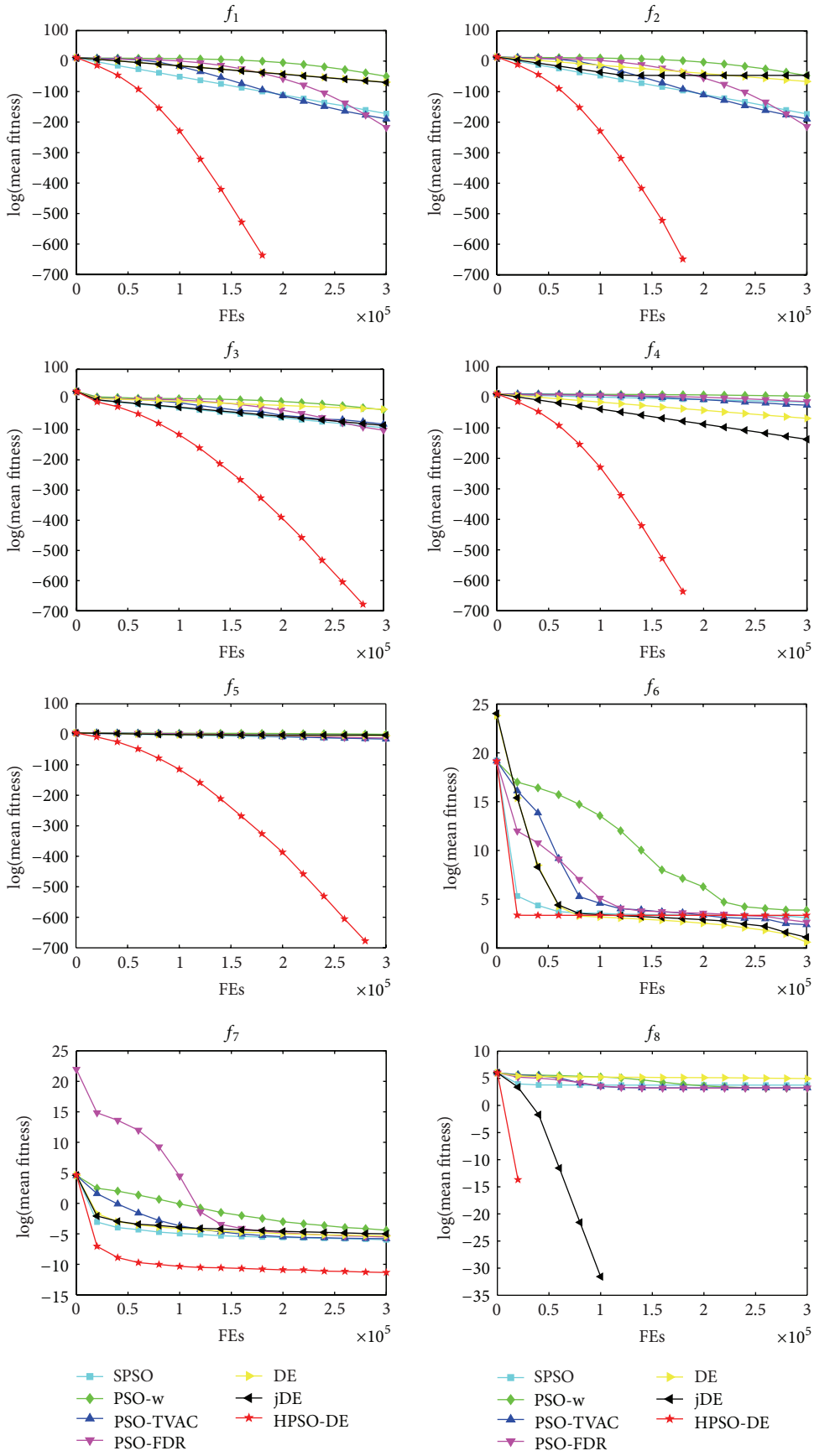

Figure 1: Performance of the algorithms for f_1, f_2, f_3, f_4, f_5, f_6, f_7, and f_8.

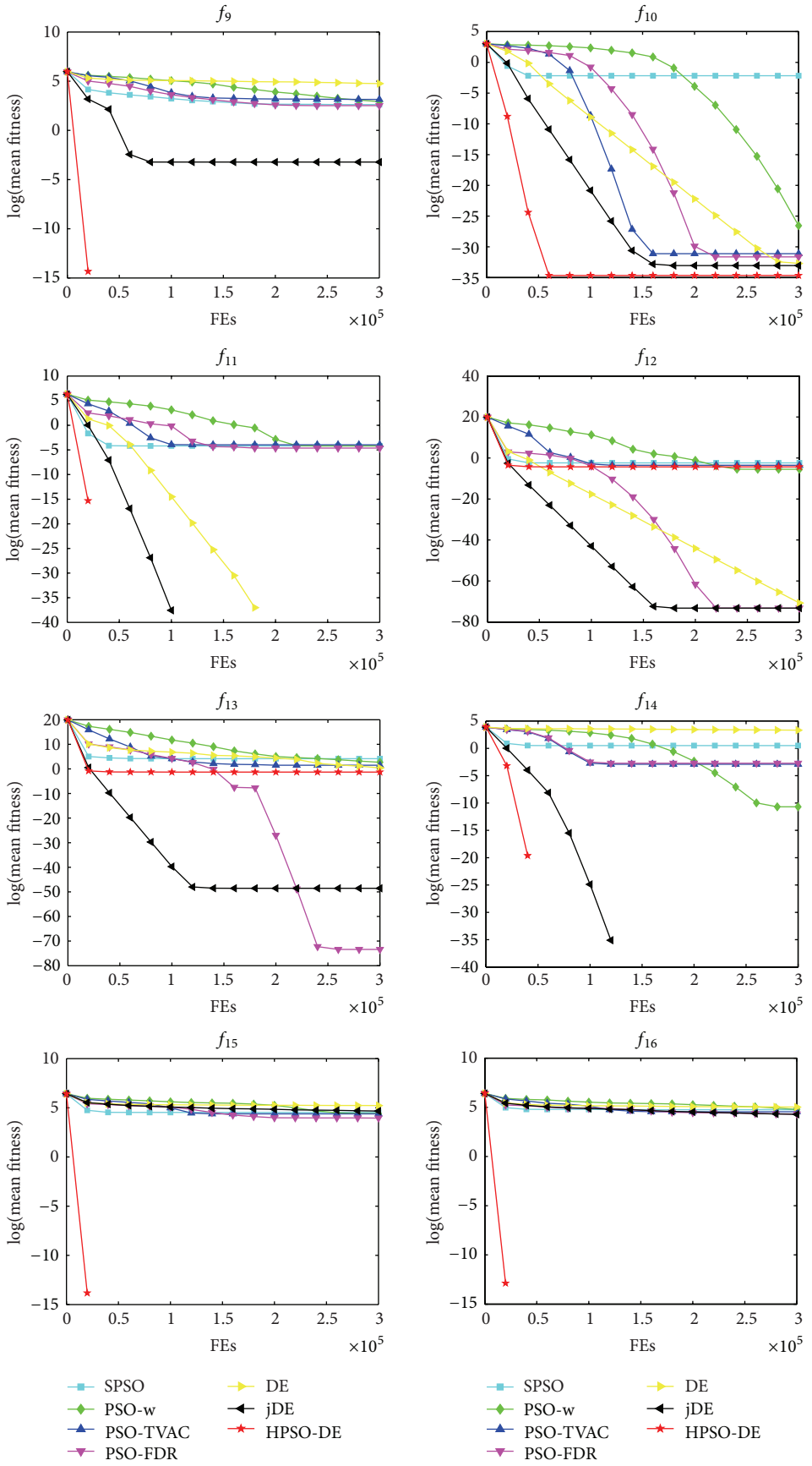

Figure 2: Performance of the algorithms for f_9, f_{10} f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, and f_{16}.

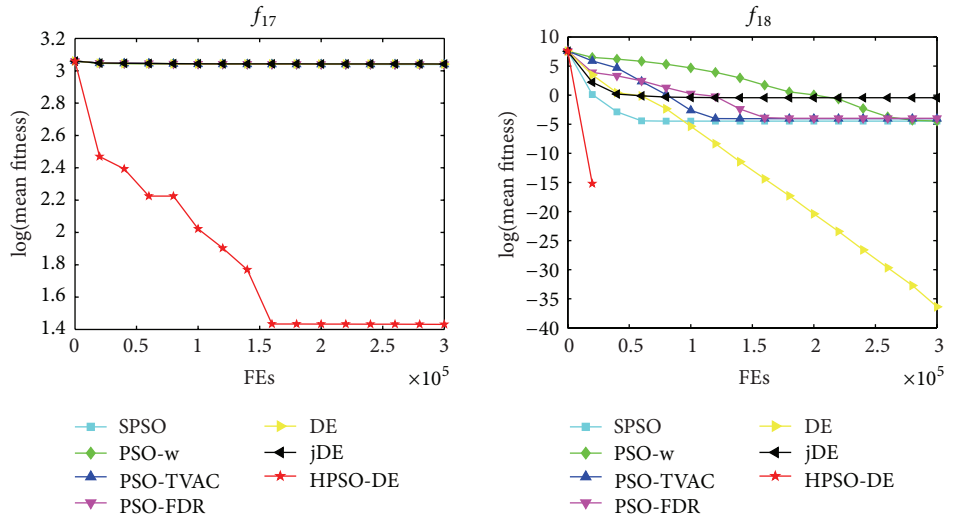

Figure 3: Performance of the algorithms for f_{17} and f_{18}.

Table 1: Benchmark configurations.

Function	Name	Search space	Global optimal $f(x^*)$	x^*
f_1	Sphere	$[-100, 100]$	0	0
f_2	Weighted sphere	$[-100, 100]$	0	0
f_3	Schwefel's Problem 2.22	$[-10, 10]$	0	0
f_4	Shifted Schwefel's Problem 1.2	$[-100, 100]$	0	0
f_5	Schwefel's Problem 2.21	$[-100, 100]$	0	0
f_6	Rosenbrock	$[-30, 30]$	0	$(1, 1, \ldots, 1)$
f_7	Quartic	$[-1.28, 1.28]$	0	0
f_8	Rastrigin	$[-5, 5]$	0	0
f_9	Noncontinuous Rastigin	$[-5, 5]$	0	0
f_{10}	Ackley	$[-32, 32]$	0	0
f_{11}	Griewank	$[-600, 600]$	0	0
f_{12}	Penalized 1	$[-50, 50]$	0	0
f_{13}	Penalized 2	$[-50, 50]$	0	0
f_{14}	Weierstrass	$[-0.5, 0.5]$	0	0
f_{15}	Rotated Rastrigin	$[-5, 5]$	0	0
f_{16}	Rotated noncontinuous Rastrigin	$[-5, 5]$	0	0
f_{17}	Rotated Ackley	$[-32, 32]$	0	0
f_{18}	Rotated Griewank	$[-600, 600]$	0	0

Table 2: Algorithms initialization.

Algorithm	Parameters	Reference
SPSO	$w = 0.729$, $c_1 = c_2 = 1.49$	[41]
PSO-w	$w = 0.9 : 0.4$, $c_1 = c_2 = 2$	[17]
PSO-TVAC	$w = 0.9 : 0.4$, $c_1 = 2.5 : 0.5$, $c_2 = 0.5 : 2.5$	[16, 38]
PSO-FDR	$c_1 = c_2 = 1$, $c_3 = 2$	[42]
DE	$F = 0.5$, CR $= 0.9$	[20, 43]
jDE	$F_l = 0.1$, $F_u = 0.9$, $\tau_1 = \tau_2 = 0.1$	[29]
HPSO-DE	$\text{PSO}_p = 0.3$, $\text{DE}_p = 0.01$, $d_c = 1.5$, $p = 0$	This paper

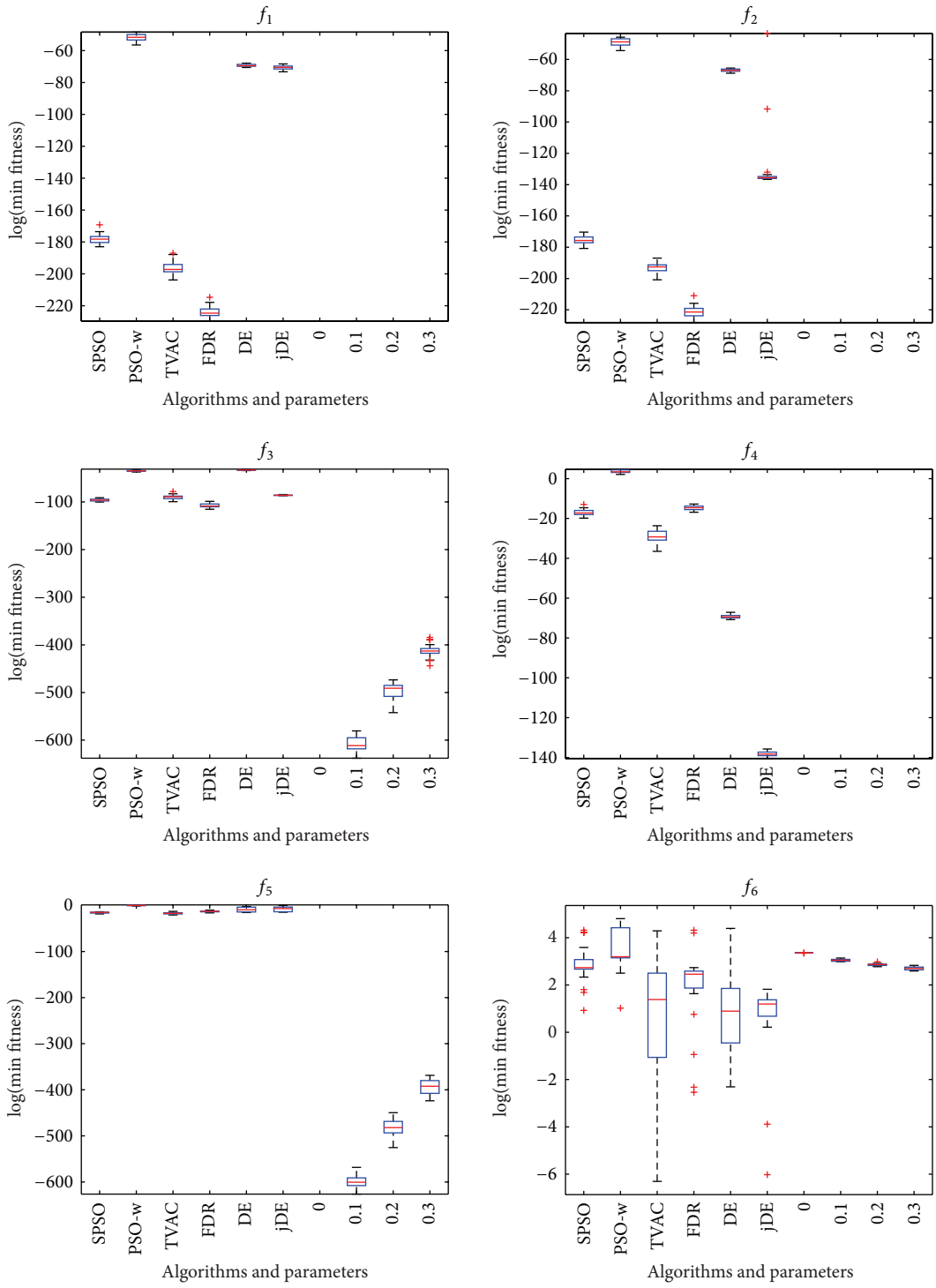

Figure 4: HPSO-DE with different balance parameters and other six algorithms on f_1, f_2, f_3, f_4, f_5, and f_6.

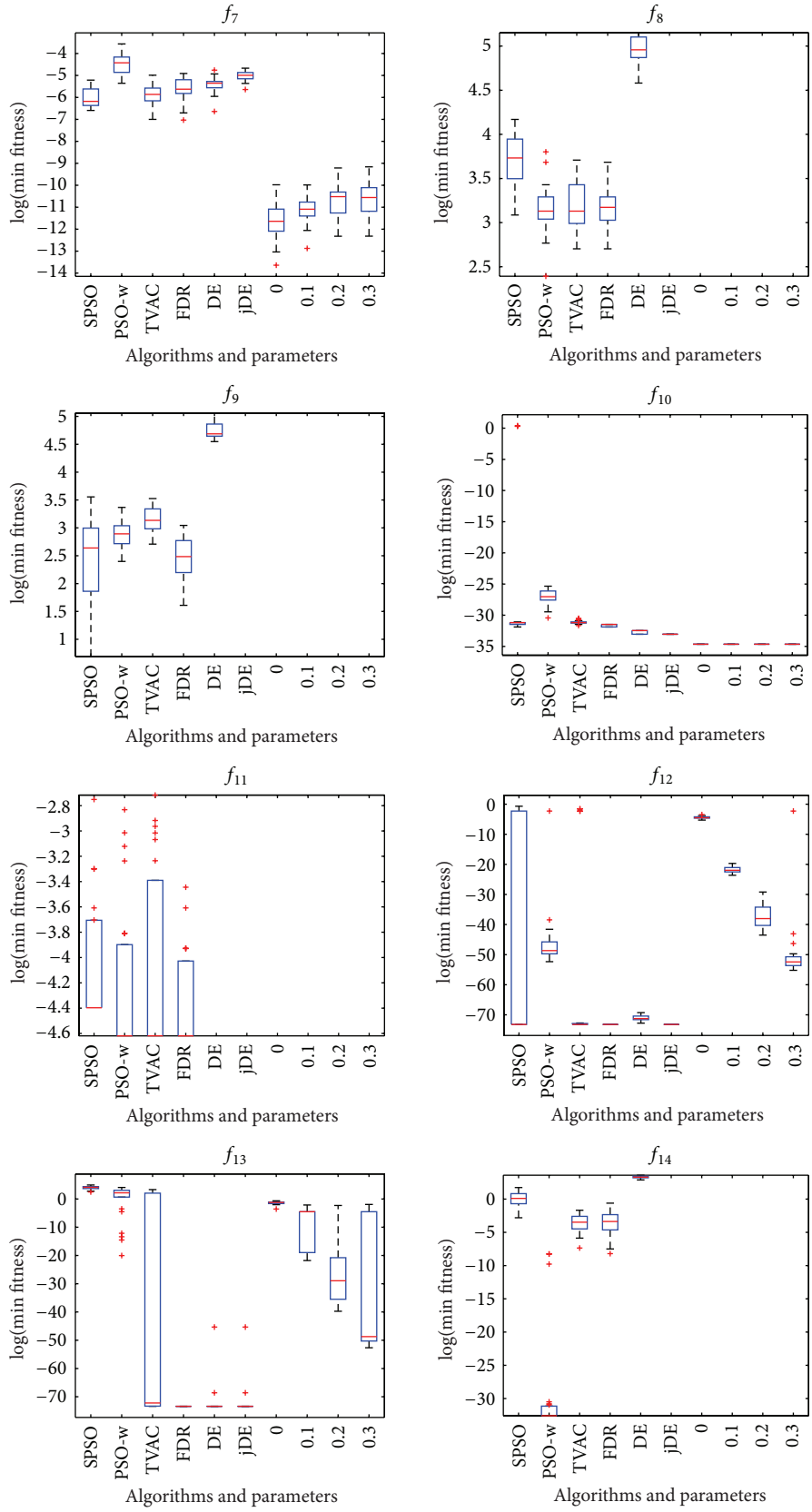

Figure 5: HPSO-DE with different balance parameters and other six algorithms on f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}, and f_{14}.

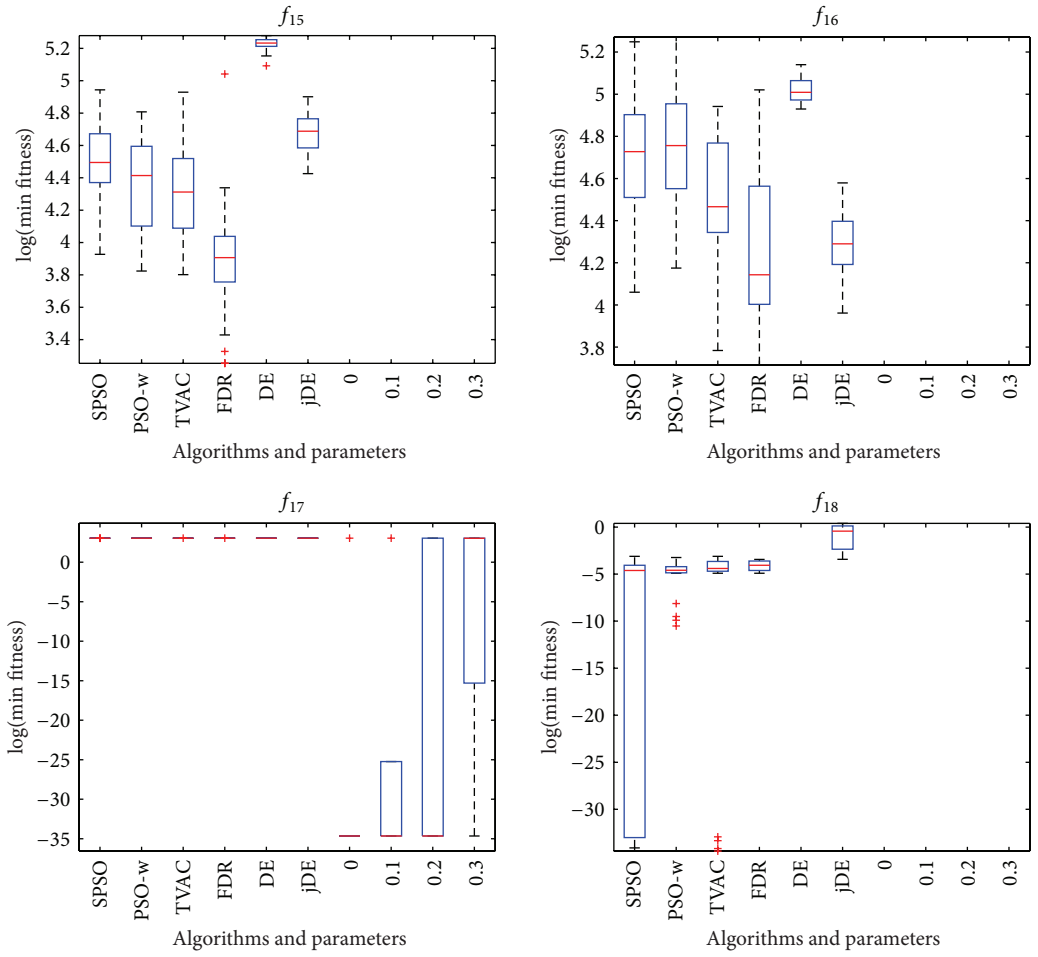

Figure 6: HPSO-DE with different balance parameters and other six algorithms on f_{15}, f_{16}, f_{17}, and f_{18}.

$$f_{10} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right) + 20 + e,$$

$$f_{11} = \frac{1}{4000}\sum_{i=1}^{D}x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{i^{1/2}}\right) + 1,$$

$$f_{12}(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{D}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{D}u(x_i,10,100,4),$$

$$y_i = \left(1+\frac{1}{4}(x_i+1)\right),$$

$$u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-a)^m, & x_i < -a, \end{cases}$$

$$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\right\} + \sum_{i=1}^{D}u(x_i,10,100,4),$$

Table 3: Result comparisons among seven algorithms on test functions.

Functions		Algorithms						
		PSO	PSO-w	PSO-FDR	PSO-TVAC	DE	jDE	HPSO-DE
f_1	Mean	$1.31E-75$	$1.61E-22$	$2.39E-95$	$4.14E-83$	$1.04E-30$	$3.84E-31$	**0.0E + 00**
	Std.	$4.03E-149$	$7.47E-44$	$1.29E-188$	$1.62E-164$	$6.81E-61$	$2.14E-61$	**0.0E + 00**
f_2	Mean	$7.44E-76$	$2.80E-21$	$9.28E-94$	$4.76E-83$	$1.33E-29$	$5.88E-21$	**0.0E + 00**
	Std.	$4.35E-150$	$1.47E-41$	$2.09E-185$	$2.0E-164$	$1.36E-58$	$8.63E-40$	**0.0E + 00**
f_3	Mean	$2.50E-41$	$1.99E-15$	$6.10E-45$	$3.88E-36$	$7.51E-15$	$5.57E-038$	**0.0E + 00**
	Std.	$3.36E-81$	$1.15E-29$	$6.64E-88$	$3.68E-70$	$5.05E-29$	$1.45E-75$	**0.0E + 00**
f_4	Mean	$1.83E-7$	$4.70E+1$	$6.77E-7$	$5.74E-12$	$1.17E-30$	$1.80E-60$	**0.0E + 00**
	Std.	$2.21E-13$	$9.39E+2$	$5.42E-13$	$1.80E-22$	$1.92E-60$	$6.01E-120$	**0.0E + 00**
f_5	Mean	$2.63E-7$	$1.0E+0$	$4.26E-6$	$1.91E-7$	$1.59E-2$	$5.92E-2$	**0.0E + 00**
	Std.	$5.79E-14$	$3.0E-1$	$5.12E-11$	$2.52E-13$	$8.59E-4$	$3.31E-2$	**0.0E + 00**
f_6	Mean	$2.24E+1$	$4.89E+1$	$1.41E+1$	$1.10E+1$	$1.10E+1$	**3.04E + 0**	$2.86E+2$
	Std.	$3.69E+2$	$1.53E+3$	$3.14E+2$	$3.50E+2$	$5.09E+2$	$2.39E+0$	**9.0E − 3**
f_7	Mean	$2.0E-3$	$1.2E-2$	$4.0E-3$	$3.0E-3$	$4.0E-3$	$7.0E-3$	**1.2E − 5**
	Std.	$1.16E-6$	$2.60E-5$	$3.18E-6$	$2.01E-6$	$2.33E-6$	$1.70E+0$	**1.049E − 10**
f_8	Mean	$4.31E+1$	$2.44E+1$	$2.45E+1$	$2.57E+1$	$1.42E+2$	**0.0E + 00**	**0.0E + 00**
	Std.	$1.31E+2$	$4.94E+1$	$3.27E+1$	$6.10E+1$	$5.11E+2$	**0.0E + 00**	**0.0E + 00**
f_9	Mean	$1.37E+1$	$1.86E+1$	$1.24E+1$	$2.33E+1$	$1.16E+2$	$4.0E-2$	**0.0E + 00**
	Std.	$7.75E+1$	$1.90E+1$	$1.61E+1$	$2.43E+1$	$2.82E+2$	$4.0E-2$	**0.0E + 00**
f_{10}	Mean	$1.14E-1$	$2.92-E12$	$1.82E-14$	$3.08E-14$	$6.29E-15$	$4.44E-15$	**8.882E − 16**
	Std.	$1.55E-1$	$6.88E-24$	$1.30E-29$	$8.60E-29$	$3.28E-30$	**0.0E + 00**	**0.0E + 00**
f_{11}	Mean	$1.5E-2$	$1.5E-2$	$1.0E-2$	$1.9E-2$	**0.0E + 00**	**0.0E + 00**	**0.0E + 00**
	Std.	$2.44E-4$	$3.0E-4$	$8.97E-5$	$4.40E-4$	**0.0E + 00**	**0.0E + 00**	**0.0E + 00**
f_{12}	Mean	$9.1E-2$	$4.0E-3$	**1.57E − 32**	$2.90E-2$	$1.87E-31$	$1.58E-32$	$1.3E-2$
	Std.	$3.2E-2$	$4.0E-4$	**3.12E − 95**	$4.0E-3$	$3.39E-62$	**3.12E − 95**	$3.30E-5$
f_{13}	Mean	$6.34E+1$	$1.45E+1$	**1.35E − 32**	$4.44E+0$	$1.49E+0$	$8.33E-22$	$2.76E-1$
	Std.	$1.07E+3$	$2.13E+2$	**3.12E − 95**	$5.26E+1$	$1.68E+1$	$1.73E-41$	$1.2E-2$
f_{14}	Mean	$1.67E+0$	$2.28E-5$	$6.65E-2$	$5.5E-2$	$2.79E+1$	**0.0E + 00**	**0.0E + 00**
	Std.	$2.03E+0$	$5.10E-9$	$1.18E-2$	$3.0E-3$	$2.78E+1$	**0.0E + 00**	**0.0E + 00**
f_{15}	Mean	$9.25E+1$	$8.24E+1$	$5.23E+1$	$8.01E+1$	$1.86E+2$	$1.07E+2$	**0.0E + 00**
	Std.	$4.84E+2$	$5.01E+2$	$5.99E+2$	$6.77E+2$	$5.82E+1$	$1.97E+2$	**0.0E + 00**
f_{16}	Mean	$1.17E+2$	$1.20E+2$	$7.52E+1$	$9.38E+1$	$1.52E+2$	$7.29E+1$	**0.0E + 00**
	Std.	$1.33E+3$	$1.00E+3$	$8.81E+2$	$7.02E+2$	$9.76E+1$	$1.28E+2$	**0.0E + 00**
f_{17}	Mean	$2.09E+1$	$2.09E+1$	$2.09E+1$	$2.09E+1$	$2.09E+1$	$2.10E+1$	**4.18E + 0**
	Std.	$2.6E-3$	$4.3E-3$	$2.2E-3$	$2.1E-3$	**1.80E − 3**	$2.2E-3$	$7.29E+1$
f_{18}	Mean	$1.15E-2$	$1.17E-2$	$1.87E-2$	$1.76E-2$	$1.60E-16$	$6.47E-1$	**0.0E + 00**
	Std.	$1E-4$	$1E-4$	$1E-4$	$2E-4$	$2.06E-31$	$2.74E-1$	**0.0E + 00**

$$f_{14} = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{\max}} \left[a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right)\right] - D \sum_{k=0}^{k_{\max}} \left[a^k \cos\left(2\pi b^k * 0.5\right)\right] \right),$$

$$a = 0.5, \quad b = 3, \quad k_{\max} = 20,$$

$$f_{15} = \sum_{i=1}^{D} \left[y_i^2 - 10\cos\left(2\pi y_i\right) + 10\right], \quad y = M \times x,$$

$$f_{16}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right),$$

$$z_i = \begin{cases} y_i, & |y_i| < 0.5, \\ \dfrac{\text{round}\left(2y_i\right)}{2}, & |y_i| \geq 0.5, \end{cases} \quad y = M \times x,$$

$$f_{17} = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos\left(2\pi y_i\right) \right) + 20 + e, \quad y = M \times x,$$

$$f_{18} = \frac{1}{4000}\sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D} \cos\left(\frac{y_i}{i^{1/2}}\right) + 1, \quad y = M \times x. \tag{19}$$

Table 4: Convergence speed and algorithm reliability comparisons.

Functions		Algorithms						
		PSO	PSO-w	PSO-FDR	PSO-TVAC	DE	jDE	HPSO-DE
f_1	FESS	4.5056E + 4	2.3642 + E5	1.4738E + 5	1.0142E + 5	1.1055E + 5	1.0883E + 5	**2.2075E + 4**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_2	FESS	4.9428E + 4	2.4217E + 5	1.5194E + 5	1.0435E + 5	1.1905E + 5	6.4498E + 4	**2.3002E + 4**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_3	FESS	6.8336E + 4	2.5089E + 5	1.6382E + 5	1.1524E + 5	1.8979E + 5	7.6644E + 4	**3.2762E + 4**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_4	FESS	2.907E + 5	—	—	2.4757E + 5	1.1041E + 5	5.9383E + 4	**2.2075E + 4**
	SR	12%	0%	0%	**100%**	**100%**	**100%**	**100%**
f_5	FESS	2.987E + 5	—	—	2.8725E + 5	—	—	**3.2629E + 4**
	SR	4%	0%	0%	24%	0%	0%	**100%**
f_8	FESS	—	—	—	—	—	7.3477E + 4	**2.3084E + 4**
	SR	0%	0%	0%	0%	0%	100%	**100%**
f_9	FESS	—		—	—	—	8.7133E + 4	**2.2356E + 4**
	SR	0%		0%	0%	0%	96%	**100%**
f_{10}	FESS	7.337E + 4	2.7072E + 5	1.7226E + 5	1.2118E + 5	1.7113E + 5	9.0429E + 4	**3.2502E + 4**
	SR	92%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
f_{11}	FESS	4.6325E + 4	2.3907E + 5	1.5296E + 5	1.0411E + 5	1.1398E + 5	6.2468E + 4	2.0659E + 4
	SR	32%	28%	36%	36%	**100%**	**100%**	**100%**
f_{12}	FESS	4.6978E + 4	2.4440E + 5	1.3834E + 5	1.0872E + 5	1.0199E + 5	**5.0201E + 4**	—
	SR	72%	96%	**100%**	80%	**100%**	**100%**	0%
f_{13}	FESS	—	2.975E + 5	1.6684E + 5	1.4981E + 5	2.5882E + 5	**5.7271E + 4**	—
	SR	0%	4%	**100%**	60%	52%	**100%**	0%
f_{14}	FESS	—	2.7346E + 5	—	—	—	8.435E + 4	**3.85E+4**
	SR	0%	88%	0%	0%	0%	**100%**	**100%**
f_{15}	FESS	—	—	—	—	—	—	**2.1025E + 4**
	SR	0%	0%	0%	0%	0%	0%	**100%**
f_{16}	FESS	—	—	—	—	—	—	**2.2563E + 4**
	SR	0%	0%	0%	0%	0%	0%	**100%**
f_{17}	FESS	—	—	—	—	—	—	**6.8878E + 4**
	SR	0%	0%	0%	0%	0%	0%	**80%**
f_{18}	FESS	1.0931E + 5	—	—	1.4448E + 5	1.834E + 5	—	**2.268E + 4**
	SR	36%	0%	0%	20%	100%	0%	**100%**

4.2. Algorithms for Comparison. Experiments are conducted on a suite of 16 numerical functions to evaluate seven algorithms including the proposed HPSO-DE algorithm. For functions, 30-dimensional (30D) function is tested. The maximum number of function evaluations (FEs) is set to 300 000 and NP is 100. All experiments are run 25 times independently. The seven algorithms in comparison are listed in Table 2.

4.3. Comparisons on the Solution Accuracy. The mean and standard deviation (Std) of the solutions in 25 independent runs are listed in Table 3. The best result among these algorithms is indicated by boldface in the table. Figures 1, 2, and 3 show the comparisons in terms of convergence, mean solutions, and evolution processes in solving 16 benchmark functions.

From Table 3 and Figures 1–3, it is very clear that the hybrid proposed algorithm has the strong ability to jump out of the local optima. It can effectively prevent the premature convergence and significantly enhance the convergence rate and accuracy. It provides best performance on the f_1, f_2, f_3, f_4, f_5, f_8, f_9, f_{11}, f_{14}, f_{15}, f_{16}, and f_{18}, which reachse the highest accuracy on them. The jDE ranks the second on f_8, and f_{11} and performs a little better than HPSO-DE on f_6. PSO-FDR performs best on f_{12}, f_{13}.

One can observe that the proposed method can search the optimum and maintain a higher convergence speed. The capabilities of avoiding local optima and finding global optimum of these functions indicate the superiority of HPSO-DE.

4.4. Comparisons on Convergent Rate and Successful Percentage. The convergent rate for achieving the global optimum is another key point for testing the algorithm performance. The success of an algorithm means that this algorithm can result

Table 5: Results of the comparison between HPSO-DE and JADE.

Functions	Gen	HPSO-DE Mean (Std Dev.)	JADE w/o archive Mean (Std Dev.)	JADE with archive Mean (Std Dev.)
f_1	1500	**2.9E − 121 (1.0E − 240)**	1.8E − 60 (8.4E − 60)	1.3E − 54 (9.2E − 54)
f_3	2000	**1.7E − 78 (4.5E − 155)**	1.8E − 25 (8.8E − 25)	3.9E − 22 (2.7E − 21)
f_4	5000	**0.0E + 00 (0.0E + 00)**	5.7E − 61 (2.7E − 60)	6.0E − 87 (1.9E − 86)
f_5	5000	**1.7E − 185 (00.0E + 00)**	8.2E − 24 (4.0E − 23)	4.3E − 66 (1.2E − 65)
f_6	3000	2.9 + E2 (1.1E − 2)	**8.0E − 02 (5.6E − 01)**	3.2E − 01 (1.1E + 00)
	20000	2.9 + E2 (3.4E − 2)	**8.0E − 02 (5.6E − 01)**	3.2E − 01 (1.1E + 00)
f_7	3000	**2.1E − 5 (2.2E − 10)**	6.4E − 04 (2.5E − 04)	6.8E − 04 (2.5E − 04)
f_8	1000	**0.0E + 00 (0.0E + 00)**	1.0E − 04 (6.0E − 05)	1.4E − 04 (6.5E − 05)
	5000	**0.0E + 00 (0.0E + 00)**	**0.0E + 00 (0.0E + 00)**	**0.0E + 00 (0.0E + 00)**
f_{10}	500	**8.9E − 16 (0.0E + 00)**	8.2E − 10 (6.9E − 10)	3.0E − 09 (2.2E − 09)
	2000	**8.9E − 16 (0.0E + 00)**	4.4E − 15 (0.0E + 00)	4.4E − 15 (0.0E + 00)
f_{11}	500	**0.0E + 00 (0.0E + 00)**	9.9E − 08 (6.0E − 07)	2.0E − 04 (1.4E − 03)
	3000	**0.0E + 00 (0.0E + 00)**	**0.0E + 00 (0.0E + 00)**	2.0E − 04 (1.4E − 03)
f_{12}	500	1.8E − 2 (1E − 4)	**4.6E − 17 (1.9E − 16)**	3.8E − 16 (8.3E − 16)
	1500	1.7E − 2 (4E − 4)	**1.6E − 32 (5.5E − 48)**	**1.6E − 32 (5.5E − 48)**
f_{13}	500	4.4E − 1 (2.1E − 2)	**2.0E − 16 (6.5E − 16)**	1.2E − 15 (2.8E − 15)
	1500	3.1E − 1 (1.6E − 2)	**1.4E − 32 (1.1E − 47)**	**1.4E − 32 (1.1E − 47)**

in a function value not worse than the prespecified optimal value, that is, for all problems with the number of function evaluations less than the pre specified maximum number. The success rate (SR) is calculated as the number of successful runs divided by the total number of runs.

In Table 4, we summarize the SR of each algorithm and the average number of function evaluations over successful runs (FESS). An experiment is considered as successful if the best solution is found with sufficient accuracy: 10^{-8}.

Table 4 shows that HPSO-DE needs least FESS to achieve the acceptable solution on most of functions, which reveals that proposed algorithm has a higher convergent rate than other algorithms. DE and jDE outperform HPSO-DE on the f_{12} and f_{13}; SPSO, LPSO, and PSO-TVAC have much worse SR and accuracy than HPSO-DE on the test functions. In addition, HPSO-DE can achieve accepted value with a good convergence speed and accuracy on most of the functions, as seen from Figures 1–3 and Table 3.

In summary, the HPSO-DE performs best on functions and has good search ability. Owing to the proposed techniques, the HPSO-DE processes capabilities of fast convergence speed, the highest successful rate, and the best search accuracy among these algorithms.

4.5. Parameter Study. The balanced parameter p needs to be optimized. In this section, we investigate the impact of this parameter on HPSO-DE. The HPSO-DE algorithm runs 25 times on each function with four different balanced parameters of 0, 0.1, 0.2, and 0.3. The influence of balanced parameters on accuracy of HPSO-DE algorithm is investigated by comparing the optima values that HPSO-DE obtains for different balanced parameters.

Figures 4, 5, and 6 show the box plots of minimal values that HPSO-DE obtains with four different balanced parameters. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the remaining data. Outliers are data with values beyond the ends of the whiskers.

From Figures 4–6, one can observe that the accuracy of HPSO-DE is less sensitive to the balanced parameter on most of functions except f_{12}, f_{13}, and f_{17} when balanced parameter is between 0 and 0.3.

4.6. Comparison with JADE. The JADE algorithm is tested on a set of standard test functions in [31]. HPSO-DE is compared with JADE on 30D test functions chosen from [31]. The parameter settings are the same as in [31]. Maximum generations are listed in Table 5. The middle results of 50 independent runs are summarized in the table (results for JADE are taken from [31]), which show that the proposed algorithm obviously performs better than the JADE algorithm.

5. Conclusions

In this paper, a novel algorithm HPSO-DE is proposed by developing a balanced parameter between PSO and DE. The population is generated either by DE or by PSO according to the balanced parameter. Adaptive mutation is carried out to current population when the population clusters around local optima. The strategy makes HPSO-DE have the advantages of two algorithms and maintain diversity of the population. In comparison with the PSO, DE, and their variants, the

proposed algorithm is more effective in obtaining better quality solutions, works in a more effective way, and finds better quality solutions more frequently.

Conflict of Interests

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

Acknowledgments

This work was supported by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), Social Science Foundation of Chinese Ministry of Education (nos. 12YJC630271 and 13YJC630018), Natural Science Fund for Colleges and Universities in Jiangsu Province (no. 13KJB120008), China Natural Science Foundation (nos. 71273139, 71101073 and 71173116) and China Institute of Manufacturing Development (no. SK20130090-15).

References

[1] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[2] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, December 1995.

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

[5] 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.

[6] Y. Tang, Z. Wang, H. Gao, S. Swift, and J. Kurths, "A constrained evolutionary computation method for detecting controlling regions of cortical networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 6, pp. 1569–1581, 2012.

[7] Y. Tang, Z. Wang, and J. Fang, "Controller design for synchronization of an array of delayed neural networks using a controllable probabilistic PSO," *Information Sciences*, vol. 181, no. 20, pp. 4715–4732, 2011.

[8] Y. Tang, Z. Wang, and J. Fang, "Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2523–2535, 2011.

[9] S. Y. S. Leung, Y. Tang, and W. K. Wong, "A hybrid particle swarm optimization and its application in neural networks," *Expert Systems with Applications*, vol. 39, no. 1, pp. 395–405, 2012.

[10] Y. P. Chen, W. C. Peng, and M. C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 37, no. 6, pp. 1460–1470, 2007.

[11] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[12] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 81–86, Seoul, Republic of Korea, May 2001.

[13] 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.

[14] 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.

[15] R. A. Krohling and L. dos Santos Coelho, "Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 36, no. 6, pp. 1407–1416, 2006.

[16] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[17] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, pp. 1945–1950, IEEE Press, Piscataway, NJ, USA, 1999.

[18] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming VII (LNCS '98)*, pp. 591–600, Springer, New York, NY, USA, 1998.

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

[20] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[21] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," in *Proceedings of the 8th International Conference on Soft Computing (MENDEL '02)*, pp. 11–18, 2002.

[22] Y. Tang, H. Gao, W. Zou, and J. Kurths, "Identifying controlling nodes in neuronal networks in different scales," *PLoS ONE*, vol. 7, no. 7, Article ID e41375, 2012.

[23] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1679–1696, 2011.

[24] S. Das and A. Konar, "Automatic image pixel clustering with an improved differential evolution," *Applied Soft Computing Journal*, vol. 9, no. 1, pp. 226–236, 2009.

[25] R. Storn, "Differential evolution design of an IIR-filter," in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 268–273, May 1996.

[26] M. Varadarajan and K. S. Swarup, "Differential evolution approach for optimal reactive power dispatch," *Applied Soft Computing Journal*, vol. 8, no. 4, pp. 1549–1561, 2008.

[27] K. Price, R. Storn, and J. Lampinen, *Differential Evolution—A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[28] R. Mallipeddi and P. N. Suganthan, "Differential evolution algorithm with ensemble ofparameters and mutation and crossover strategies," in *Proceedings of the Swarm Evolutionary and Memetic Computing Conference*, vol. 6466, pp. 71–78, Chennai, India, 2010.

[29] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[30] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 1110–1116, June 2008.

[31] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[32] M. Pant and R. Thangaraj, "DE-PSO: a new hybrid metaheuristic for solving global optimization problems," *New Mathematics and Natural Computation*, vol. 7, no. 3, pp. 363–381, 2011.

[33] S. Kannan, S. M. R. Slochanal, P. Subbaraj, and N. P. Padhy, "Application of particle swarm optimization technique and its variants to generation expansion planning problem," *Electric Power Systems Research*, vol. 70, no. 3, pp. 203–210, 2004.

[34] H. Talbi and M. Batouche, "Hybrid particle swarm with differential evolution for multimodal image registration," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT '04)*, vol. 3, pp. 1567–1572, December 2004.

[35] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Differential evolution based particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 112–119, April 2007.

[36] Z. F. Hao, G. H. Guo, and H. Huang, "A particle swarm optimization algorithm with differential evolution," in *Proceedings of the 6th International Conference on Machine Learning and Cybernetics (ICMLC '07)*, pp. 1031–1035, August 2007.

[37] Y. Tang, Z. D. Wang, and J. A. Fang, "Feedback learning particle swarm optimization," *Applied Soft Computing Journal*, vol. 11, no. 8, pp. 4713–4725, 2011.

[38] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, "Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients," *Information Sciences*, vol. 177, no. 22, pp. 5033–5049, 2007.

[39] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Computing*, vol. 15, no. 11, pp. 2127–2140, 2011.

[40] H. Sharma, J. C. Bansal, and K. V. Arya, "Fitness based differential evolution," *Memetic Computing*, vol. 4, pp. 303–316, 2012.

[41] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, April 2007.

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

[43] M. M. Ali and A. Törn, "Population set-based global optimization algorithms: some modifications and numerical studies," *Computers and Operations Research*, vol. 31, no. 10, pp. 1703–1725, 2004.

[44] M. Ali and M. Pant, "Improving the performance of differential evolution algorithm using Cauchy mutation," *Soft Computing*, vol. 15, no. 5, pp. 991–1007, 2011.

[45] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '05)*, vol. 2, pp. 1785–1791, September 2005.