Feedback learning particle swarm optimization

Yang Tang, Zidong Wang, and Jian-an Fang

Abstract

In this paper, a feedback learning particle swarm optimization algorithm with quadratic inertia weight (FLPSO-QIW) is developed to solve optimization problems. The proposed FLPSO-QIW consists of four steps. Firstly, the inertia weight is calculated by a designed quadratic function instead of conventional linearly decreasing function. Secondly, acceleration coefficients are determined not only by the generation number but also by the search environment described by each particle's history best fitness information. Thirdly, the feedback fitness information of each particle is used to automatically design the learning probabilities. Fourthly, an elite stochastic learning (ELS) method is used to refine the solution. The FLPSO-QIW has been comprehensively evaluated on 18 unimodal, multimodal and composite benchmark functions with or without rotation. Compared with various state-of-the-art PSO algorithms, the performance of FLPSO-QIW is promising and competitive. The effects of parameter adaptation, parameter sensitivity and proposed mechanism are discussed in detail.

Index Terms

Particle swarm optimization, feedback learning, neural networks, parameters estimation

I. Introduction

PARTICLE swarm optimization (PSO) is a population-based search optimization technique introduced by Eberhart and Kennedy [1]. The PSO employs a simple method that simulates swarm behavior in agents such as fish gathering and birds flocking to guide the particles to search for the potential optimal solutions. Each particle in the swarm stands for a potential solution of the optimization problem. A particle flies to a new position according to the new velocity and its previous positions. Thus, all of the particles iteratively discover a probable solution according to the velocity and position updating equations.

Over the past decade, the PSO algorithm has gained wide-spread popularity in the research community mainly because of its simplistic implementation, reported success on benchmark test problems, and acceptable performance on various application problems. PSO algorithm has been successfully used in multiobjective optimization [2] and constrained optimization [3], etc. Unfortunately, when solving complex multimodal tasks, the conventional PSO algorithm can easily fly into the local optima and lack the ability to jump out of the local optima. These shortcomings have imposed the restrictions on the wider applications of the PSO to real-world optimization problems. Therefore, the abilities of good convergence speed and avoiding local optima are two important and appealing tasks in the study of PSO. A number of variants of PSO algorithms have been developed in order to achieve these two goals, see for example [3-24].

Firstly, the parametric studies on coefficients (inertia weight and coefficients) were conducted in [9, 10, 13-15]. The velocity term in PSO indicates a particle's ability to explore the search space while it moves under the attraction of 'pbest' and 'gbest'. In initial phases of PSO's search, the concept of linearly decreasing inertia weight with generation number was introduced in [14]. The strategy has gained

This research was partially supported by the National Natural Science Foundation of PR China (Grant No 60874113), the Research Fund for the Doctoral Program of Higher Education (Grant No 200802550007), the Key Creative Project of Shanghai Education Community (Grant No 09ZZ66), the Key Foundation Project of Shanghai(Grant No 09JC1400700), the International Science and Technology Cooperation Project of China under Grant 2009DFA32050, and the Alexander von Humboldt Foundation of Germany.

Y. Tang is with the School of Information Science and Technology, Donghua University, Shanghai 201620, China and Space Control and Inertial Technology Research Center, Harbin Institute of Technology, Harbin 150001, China, and also with Potsdam Institute for Climate Impact Research, Potsdam, Germany. e-mail: tangtany@gmail.com.

Z. Wang is with the Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K. and also with the School of Information Science and Technology, Donghua University, Shanghai 201620, China. e-mail: Zidong.Wang@brunel.ac.uk.

J. Fang is with the School of Information Science and Technology, Donghua University, Shanghai 201620, China. e-mail: jafang@dhu.edu.cn.

popularity in promoting convergence of PSO. The idea of varying coefficients was also extended to dynamically update the acceleration coefficients in [13]. Convergence and stability analyses were proposed by Clerc and Kennedy [9]. Based on the theoretical analysis, the constriction factor has been introduced into PSO to analyze the convergence behavior. Recently, it is shown that adaptive control parameters based evolutionary algorithms perform well on various benchmarks. The adaptive idea was also applied in designing variants of PSOs [16, 21]. In Ref. [16], an adaptive particle swarm optimization (APSO) was presented. By evaluating the population distribution and particle fitness, four evolutionary states are defined, which enable the automatic control of inertia weight, acceleration coefficients, and other algorithmic parameters at run time to improve the search efficiency and convergence speed. It was found that APSO performs well on unimodal and unrotated optimization problems with fast convergence speed. However, when dealing with rotated optimization problems and composite problems, APSO is confronted with premature convergence problem. In the field of adaptive tuning population size, an efficient population utilization strategy for PSO (EPUS-PSO) was presented [21], adopting an adaptive population manager to improve the efficiency of PSO.

Secondly, according to the idea of graph theory and network theory, some network based PSOs have been proposed [7, 11, 12, 17-20, 24]. In Ref. [18], several social network structures were tested, with small-world randomization of a specified number of links. Furthermore, in Ref. [11], a fully informed particle swarm optimization (FIPSO) was proposed and several topologies of swarm were tested. In Ref. [20], it was found that the graphs in FIPSO which performed well were more highly clustered. The inhibition of communication might contribute to the discovery of global optima in the FIPS algorithm, while promotion of communication speeds convergence within optimal regions. By the idea of changing learning (communication) strategies in swarm, a comprehensive learning PSO (CLPSO) was proposed [12], where the learning strategy abandons the global best information, and all other particles' past best positions might be used to update particles' velocity. Although CLPSO delivers good performance on complex multimodal functions, its convergence speed is not satisfactory for unimodal, multimodal and rotated optimization problems.

Thirdly, hybridization by combining PSO with other search approaches has drawn increasing attention to improve the performance of the PSO [5, 6, 7, 22, 23, 25, 26]. Evolutionary operators and other methods such as selection [6], crossover, mutation [5] and chaotic sequences [25, 26] have been introduced to the PSO to keep the best particles, to increase the diversity of the population, and to improve the capability to escape local minima. In [22], the swarm was divided into sub-populations, and a breeding operator is then used within a subpopulation or between the sub-populations to enhance the diversity of the swarm population. In [7], a cooperative particle swarm optimizer (CPSO-H) was proposed to lead to improvement of the original PSO for multimodal optimization problems. In Ref. [23], similarities between PSO and evolutionary optimization (EO) are presented. The authors suggest an evolutionary algorithm (EA) which is fundamentally equivalent to a PSO and then introduce different EA-specific operators to the constriction-based PSO. Contrary to the standard PSO algorithms, the hybrid PSO, which performs similarly to the existing genetic algorithm (GA) (and outperforms the GA in some occasions), is developed by replacing PSO's standard child-creation rule with a parent-centric recombination operator.

Although a number of works can help to improve the search ability and enhance convergence speed of PSO, there still remains research room to improve the performance of PSO. With respect to existing benchmark algorithms, the existing PSO framework does not provide a way-out to solve such simple, unconstrained, unimodal, multimodal problems competitively. In this study, a novel PSO using fitness feedback mechanism and quadratic inertia weight is proposed, which aims to enable the PSO to gain the abilities of good convergence speed and search performance at the same time. The inertia weights is calculated according to a quadratic function. The learning probabilities are determined by fitness feedback mechanism. Moreover, acceleration coefficients are computed by the fitness feedback mechanism and generation number. In addition, stochastic disturbances controlled by adaptive probability are added only to the globally best particle under adaptive probabilities to refine the solution. The experimental results on various benchmark functions with different topological structures are confirmed to show effectiveness of FLPSO-QIW.

This paper is organized as follows. Section II describes the feedback learning PSO with a quadratic function. Section III presents the test functions, the experimental setting for each algorithm. Concluding remarks are given in Section IV.

II. FLPSO-QIW

A. PSO algorithm

A swarm consists of N particles moving around in a D-dimensional search space. The position of the ith particle is written by a vector $x_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{iD}(k))$, where $x_{ij}(k) \in [x_{\min,j}, x_{\max,j}], 1 \leq j \leq D, x_{\min,j}$ and $x_{\max,j}$ are lower and upper bounds for the jth dimension respectively. The particle tunes its current position toward the global optimum according to two terms: the best position encountered by itself (*pbest*) represented by $p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ and the best position in the whole swarm (*gbest*) represented by $p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$. The velocity of the ith particle at the kth iteration is represented by $v_i(k) = (v_{i1}(k), v_{i2}(k), \cdots, v_{iD}(k))$, which is limited to a maximum velocity $v_{\max} = (v_{\max,1}, v_{\max,2}, \cdots, v_{\max,D})$. Thus, if the magnitude of the updated velocity $|v_{i,d}|$ exceeds $v_{\max,d}$, then $v_{i,d}$ is assigned the value $sign(v_{i,d})v_{\max,d}$. In this paper, the maximum velocity $v_{\max,d}$ is set to 20% of the search range. The inertia weight w is a scaling factor controlling the influence of the old velocity on the new one; $r_{1,j}$ and $r_{2,j}$ are two uniform random number samples from $U(0,1)$. The parameters c_1 and c_2 are called acceleration coefficients, namely *cognitive* and *social* coefficients respectively. The velocity and position of the particle at next step changes according to the following equations:

$$\begin{aligned} v_{i,j}(k+1) &= w(k)v_i(k) + c_1 r_{1,j}(p_{i,j}(k) - x_{i,j}(k)) + c_2 r_{2,j}(p_{g,j}(k) - x_{i,j}(k)), \\ x_{i,j}(k+1) &= x_{i,j}(k) + v_{i,j}(k+1). \end{aligned} \tag{1}$$

Although PSO algorithms have been used to solve many optimization problems and numerous PSOs have been presented, solving complex optimization problems with both high accuracy and rapid convergence speed is still a challenging issue and remains open. In this section, a feedback learning PSO with quadratic inertia function is proposed.

B. Position and velocity updating equations of FLPSO-QIW

The positions and velocity updating equations of FLPSO-QIW are given as follows:

$$\begin{aligned} v_{i,j}(k+1) \;\; = \;\; & w(k)v_{i,j}(k) + \alpha_i(k)c_i(k)r_j(k)(p_{r,j}(k) - x_{i,j}(k)) \\ & + (1 - \alpha_i(k))c_i(k)r_j(k)(p_{i,j}(k) - x_{i,j}(k)), \end{aligned} \tag{2}$$

$$x_{i,j}(k+1) = x_{i,j}(k) + v_{i,j}(k+1), \tag{3}$$

where $p_{r,j}(k)$ represents another particles's history best fitness in the jth dimension [12]; $w(k)$ is the time-varying inertia weight determined by generation number; $c_i(k)$ is the acceleration coefficient which is time-varying according to the search circumstances. $\alpha_i(k)$ is stochastic variable that describes the following random events for the system (2):

$$\begin{cases} \text{Event 1:} & \text{system (2) experiences } p_{r,j}(k), \\ \text{Event 2:} & \text{system (2) experiences } p_{i,j}(k), \end{cases} \tag{4}$$

Let $\alpha_i(k)$ be Bernoulli distributed sequences defined by

$$\alpha_i(k) = \begin{cases} 1, & \text{if Event 1 occurs,} \\ 0, & \text{if Event 2 occurs,} \end{cases} \tag{5}$$

where $\alpha_i(k)$ satisfies Prob$\{\alpha_i(k) = 1\} = \alpha_{i0}$, Prob$\{\alpha_i(k) = 0\} = 1 - \alpha_{i0}$. α_{i0} denotes the learning probability rate. In order to pay more attention to α_{i0}, the velocity updating equation is written in the form of (2). The stochastic variable α_{i0} describes different learning strategies occurring in a probabilistic way. Note that using binary sequence switching to illustrate stochastic event is widely used in our previous works to describe some randomly occurring events, such as miss measurement, random packet losses and stochastic delay [27, 28]. Moreover, the stochastic variables can represent the missing information in a flock of birds. The randomly occurring learning methods are described by a binary switching sequence that is specified by a conditional probability distribution. It was found that different learning-probability values assigned to the particles will affect the search results [12]. However, in the proposed PSO, the learning probability is time-varying according to the search environment. The details of assigning learning probability to each particle will be illustrated in the following. In such a way, the learning probability of each particle is assigned different. For each dimension of particle i, a random number is generated. If the random number is larger than α_{i0}, the corresponding dimension will learn from its own $p_{i,j}(k)$; otherwise it will learn from another particle's $p_{i,j}(k)$.

C. Calculating inertia weight according to a quadratic function

Differently from the conventional linearly decreased inertia weight, the inertia weight in this paper is calculated according to the following quadratic function:

$$w(k) = a(k - k_{\max})^2 + b, \tag{6}$$

where a and b are the parameters to be determined. We can obtain a and b by solving the following equation:

$$\begin{cases} w_1 = a(0 - k_{\max})^2 + b, \\ w_2 = a(k_{\max} - k_{\max})^2 + b, \end{cases} \tag{7}$$

where w_1 and w_2 are the initial and final inertia weight according to the time. By solving the above equation, $w(k)$ can be calculated as follows:

$$w(k) = \frac{w_1 - w_2}{k_{\max}^2}(k - k_{\max})^2 + w_2. \tag{8}$$

In this paper, w_1 and w_2 is set to $w_1 = 0.9$ and $w_2 = 0.2$, respectively. In the section of experiments, the parameters selection and comparison with the linearly decreasing inertia weight will be discussed in detail.

D. Determining acceleration parameters and learning probability using fitness feedback

The acceleration coefficient c is usually used to balance the global and local search abilities of PSO [13]. The best reported results were achieved when c_2 starts by 0.5 increases linearly to reach 2.5 in the last iteration, and c_1 starts with 2.5 and decreases to 0.5 in the last iteration. However, only time information was taken into account in the PSO-TVAC scheme, while the search information at each step is not utilized although the fitness information might enhance the performance of PSO. Generally, a large acceleration coefficient will enable the particle to learn from other particle or its best position found so far more quickly, while a small acceleration coefficient will let the particle retain in the current position to search its near position and learn from other particle slowly. In this subsection, the control scheme acceleration coefficient will be determined by both the generation number and fitness information.

Each particle in the swarm plays a different role in the searching process. Generally speaking, if a bad particle (with bad fitness information) has a large velocity to move toward the exemplar, it would enable itself to learn from other particles and thus achieve global search rapidly. Meanwhile, if a small speed is assigned to a good particle, it would make it search around itself to refine the current search solution. In order to assign different velocity to each particle according to the fitness information, one proper way is to control acceleration coefficients. We normalize the best fitness value F_i of each particle found so far as follows:

$$G_i = \frac{F_i - F_{\min}}{F_{\max} - F_{\min}}, \quad (i = 1, 2, \cdots, N) \in [0, 1]. \tag{9}$$

where $F_{\max}$ and $F_{\min}$ are the maximum and minimum values among F_i. In this paper, G_i can be used to determine the acceleration coefficient of each particle, together with generation number. Hence, $c_i(k)$ can be calculate as follows:

$$c_1(k) = \hat{c}_1 - \frac{k}{k_{\max}}(\hat{c}_1 - \check{c}_1), \quad (i = 1, 2, \cdots, N), \tag{10}$$

$$c_2(k) = \hat{c}_2 - \frac{k}{k_{\max}}(\hat{c}_2 - \check{c}_2), \quad (i = 1, 2, \cdots, N), \tag{11}$$

$$c_i(k) = (c_2(k) - c_1(k)) * G_i + c_1(k), \quad (i = 1, 2, \cdots, N), \tag{12}$$

where $\hat{c}_1$ and $\hat{c}_2$ are the initial value for $c_1(k)$ and $c_2(k)$, respectively. $\check{c}_1$ and $\check{c}_2$ are the final value for $c_1(k)$ and $c_2(k)$, respectively. In this paper, the parameters setting are $\hat{c}_1 = 2$, $\hat{c}_2 = 1$, $\check{c}_1 = \check{c}_2 = 1.5$. By the above method, it can be observed that in the early phase, the $c_i(k)$ has a wide range at the begging and gradually turns into a small range. This mechanism can make the swarm benefit global search at the beginning and local search in the end. Furthermore, in each step, the worst particle has the largest speed to enable global search and the best particle has the smallest speed to refine the current solution. By using this linear transformation, the swarm has the acceleration coefficient belonging to the range $[c_2(k), c_1(k)]$ in the search process. Different from conventional method, both generation number and fitness feedback information are taken into account for adjusting acceleration coefficient of each particle. The acceleration coefficient can be calculated easily. Owing to our design, $c_i(k)$ is not only monotonic with time, but also monotonic with G_i. Therefore, $c_i(k)$ will have its unique value characterized by G_i and k. If a better solution is found outside the swarm, almost all the particles in the swarm will have large G_i and thus generate large $c_i(k)$. The particles will have large velocity and realize the global search. Meanwhile, the best particle away from other particles will not move dynamically until the other particles move around it. If the swarm gathers for searching solution more accurately, nearly all the particles will have small G_i and c_i, which will enable the swarm to have the local search. It is easy to implement the above method in PSO since the calculation of G_i and c_i do not need extra computation. Although the idea of using G_i to control acceleration coefficients has been proposed in [4], the control method of acceleration coefficients in this paper is different from [4]. Moreover, other modifications and improvements have been made in our algorithm.

In addition, it was found that different values of learning probability yielded different results on the same problem [12]. In order to address this problem in a general manner, the authors proposed that each particle has a different value. Therefore, particles have different levels of exploration and exploitation capability in the population and are able to solve various kinds of problems. The learning probability for each particle were empirically developed in the following [12]:

$$\alpha_{i0} = 0.05 + 0.45 * \frac{\exp(\frac{10(i-1)}{N-1}) - 1}{\exp(10) - 1}, (i = 1, 2, \cdots, N). \tag{13}$$

From the above equation, it can be seen that the learning probability of each particle is assigned according to the index i. Unfortunately, the search environment has not been taken into account. In this paper, the best fitness value F_i of each particle found so far are sorted in in an ascending way (where we suggest the best particle has the minimum fitness value) and remember the former index as I_i. The novel learning probability of each particle is assigned as follows:

$$\alpha_{I_i0} = 0.05 + 0.45 * \frac{\exp(\frac{10(i-1)}{N-1}) - 1}{\exp(10) - 1}, (i = 1, 2, \cdots, N). \tag{14}$$

An important fact to note is that the worst particle will be given the largest learning probability to learn from other particles, while the best particle will have the smallest learning probability to learn from other particle or retain the current position. Thus, by such a manner, each particle has its unique learning probability in the search process and the swarm will search more efficiently.

According to proposed learning probability, if the fitness of a particle is not updated for a refresh gap $m = 1$, a random number will be generated for each dimension of the particle. If the random number is smaller than the learning probability, event 1 will occur according to (2) and (5). According to I_i, the first 50% particles are used to generate two potential exemplars and then choose the better one as the guider. Different from the strategy in CLPSO [12], we use the fitness feedback information and good particles are utilized as potential guiders. This mechanism will help the particles to learn from right directions more efficiently and keep the diversity of population at the same time.

E. Elite stochastic learning

In this subsection, stochastic learning is introduced to the best particle for refining the solution. The elite stochastic learning (ESL) randomly selects one dimension of the best particle, which is written as $p_{g,j}(k)$ for the jth dimension. Only one dimension is considered in that the local optima are likely to have better solution in one dimension. The ESL can be written as follows:

$$p_{g,j}(k) = p_{g,j}(k) + ((\lambda_{1,j}(k) - \lambda_{2,j}(k)) * r_1(k) + \lambda_{2,j}(k)) * R_1(k), \tag{15}$$

$$p_{g,j}(k) = p_{g,j}(k) + ((\eta_1(k) - \eta_2(k)) * r_2(k) + \eta_2(k)) * R_2(k), \tag{16}$$

where $r_1(k)$ and $r_2(k)$ are uniform random numbers sample from $U(0, 1)$. $\lambda_{1,j}(k)$ and $\lambda_{2,j}(k)$ are the upper and lower bounds of the chosen particle in the swarm in jth dimension at each step, respectively. $\eta_1(k)$ and $\eta_2(k)$ are the maximal and minimal value of all dimensions of the globally best particle at each generation, respectively. $R_1(k)$ and $R_2(k)$ are the search radius of the strategy.

$$R_i(k) = \sigma(k)\delta_1 + (1 - \sigma(k))\delta_2, (i = 1, 2), \tag{17}$$

where $\delta_1 = 1$ and $\delta_2 = 0.1$. Similar to $\alpha_i(k)$, $\sigma(k)$ is a stochastic variable that describes the following random events for the system (17):

$$\begin{cases} \text{Event 1:} & \text{system (17) experiences } \delta_1, \\ \text{Event 2:} & \text{system (17) experiences } \delta_2, \end{cases} \tag{18}$$

Let $\sigma(k)$ be Bernoulli distributed sequences defined by

$$\sigma(k) = \begin{cases} 1, & \text{if Event 1 occurs,} \\ 0, & \text{if Event 2 occurs,} \end{cases} \tag{19}$$

where $\sigma(k)$ satisfies Prob$\{\sigma(k) = 1\} = \sigma_0(k)$, Prob$\{\sigma(k) = 0\} = 1 - \sigma_0(k)$. $\sigma_0(k)$ denotes the radius occurring rate. The $\sigma_0(k)$ is empirically decreased with time, which is given by

$$\sigma_0(k) = (\sigma_1 - \sigma_2) \times \frac{k_{\max} - k}{k_{\max}} + \sigma_2, \tag{20}$$

where $\sigma_1 = 1$ and $\sigma_2 = 0$. In ESL, the new position can be adopted if the fitness is better than the current globally best position. Otherwise, the new position is abandoned.

Define P_1 and P_2 as (15) activation variable and (16) activation variable, respectively. The activation probability variable P_i is updated as follows:

$$P_i = \begin{cases} P_i + \kappa_1, & \text{if the search strategy } i \\ & \text{ succeed to update the globally best particle,} \\ P_i - \kappa_2, & \text{if the search strategy } i \\ & \text{fails to improve the globally best particle,} \end{cases} \tag{21}$$

where $\kappa_1 = 0.1$ and $\kappa_2 = 0.001$; $i = 1, 2$ represents the search approach (15) and (16), respectively. Here, it is assumed that P_i ranges from $[0.1, 1]$.

In addition to the ELS, there exist five major differences with CLPSO [12] and the proposed learning methods. The main differences arise from the usage of available fitness information and five major differences can be observed as follows:

(1) Instead of using fixed acceleration coefficient c, each particle is assigned a unique value $c_i(k)$ according to fitness information feedback and generation number.

(2) Instead of using unalterable learning probability for each particle, each particle is set a unique learning probability using fitness feedback strategy.

(3) Instead of using two potential exemplars from the swarm, only the best 50% particles are used to serve as potential guiders.

(4) Instead of using the refreshing gap $m = 7$, $m = 1$ is adopted in our paper to increase the convergence rate.

(5) Instead of using linearly decreasing inertia weight, in this paper, a quadratic function is proposed to compute the inertia weight.

In summary, benefiting from the proposed techniques, the swarm will have the capabilities of fast convergence rate and keep the diversity of the population at the same time. The following experimental results will show the good performance of the proposed method.

To summarize, the pseudo code of FLPSO-QIW algorithm is described as follows by above discussion:

```
FLPSO-QIW.InitializeParameters();
FLPSO-QIW.CalInertiaWeight(); according to Eq. (8)
while ($FE$ is not equal to $FE_{\max}$)
{
for $i = 1$:particle numbers $N$
  {
  FLPSO-QIW.UpdatePosition();// update positions of particle $i$ according to Eq. (3)
  val=FLPSO-QIW.CalculateFitness();// calculate fitness of particle $i$
  FLPSO-QIW.UpdateFE();// calculate FEs
    if (val $<$ the best fitness of particle $i$ found so far)
    {
    save the position $x_i(k)$ and val;
    }
  }
FLPSO-QIW.CalLearningProb();// according to Eq. (14)
FLPSO-QIW.ESL();// according to Eq. (15)- Eq. (21)
for $i = 1$:particle numbers $N$
  {
  FLPSO-QIW.CalAccerCoef();// calculate acceleration coefficient using Eqs. (9)-(12)
  for $j = 1$:dimension $D$
    {
    FLPSO-QIW.UpdateVelocity();// update velocity according to Eqs. (2)-(5)
    }
  }
}
```

III. Experimental results

A. Experiments setup

Eighteen benchmark functions are listed in Table I and (22)-(37) are used to test the performance of PSOs [29-31]. The threshold ε in Table I is the accepted value when satisfying $|f(x)-f(x^*)|<\varepsilon$. Biased initializations are used for the functions whose global optimum is at the center of the search range. All the functions are tested on 30 dimensions. $f_1(x)$ and $f_2(x)$ are unimodal optimization problems. $f_1(x)$ is used to test the convergence speeds of PSOs. $f_2(x)$ is hard to optimize and can be viewed as a multimodal problem. $f_9(x)$ is unimodal function with rotation. $f_3(x)$ to $f_8(x)$ are multimodal problems which are hard to optimize. Note that some functions are separable and can be solved by using D one-dimensional searches. Therefore, eight rotated unimodal and multimodal problems are used to test the performance of the PSOs. According to the Salomon's method [32], 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 $y = M * x$. 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. Thus, the rotated function cannot be solved by D one-dimensional searches. f_9-f_{16} are rotated functions. f_{17} and f_{18} are composite functions with rotation. The composition functions are asymmetrical multimodal problems, with different properties in different areas [29-31].

$$f_1(x) = \sum_{i=1}^{D} x_i^2, \tag{22}$$

$$f_2(x) = \sum_{i=1}^{D-1} (100(x_{i+1}-x_i)^2 + (x_i-1)^2), \tag{23}$$

$$f_3(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10), \tag{24}$$

$$f_4(x) = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10), \quad \text{where} \quad y_i = \begin{cases} x_i, & |x_i| < 0.5, \\ \frac{round(2x_i)}{2}, & |x_i| \geq 0.5, \end{cases} \tag{25}$$

$$f_5(x) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i} + 20 + e, \tag{26}$$

$$f_6(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1, \tag{27}$$

$$f_7(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_D-1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4),$$
$$\text{where} \quad y_i = (1+\frac{1}{4}(x_i+1)), u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i-a)^m, & x_i < -a. \end{cases} \tag{28}$$

$$f_8(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i+0.5))] - D\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k * 0.5)]\right),$$
$$a=0.5, b=3, k_{\max}=20, \tag{29}$$

$$f_9(x) = \sum_{i=1}^{N}(\sum_{j=1}^{i} x_j)^2, \qquad y = M*x, \tag{30}$$

$$f_{10}(x) = \sum_{i=1}^{D-1}(100(y_{i+1}-y_i)^2 + (y_i-1)^2), \qquad y = M*x, \tag{31}$$

$$f_{11}(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i)+10), \qquad y = M*x, \tag{32}$$

$$f_{12}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i)+10),$$
$$\text{where} \quad z_i = \begin{cases} y_i, & |y_i| < 0.5, \\ \frac{round(2y_i)}{2}, & |y_i| \geq 0.5, \qquad y = M*x, \end{cases} \tag{33}$$

$$f_{13}(x) = -20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi y_i} + 20 + e, \tag{34}$$

$$f_{14}(x) = \frac{1}{4000}\sum_{i=1}^{D} y_i^2 - \prod_{i=1}^{D}\cos(\frac{y_i}{\sqrt{i}}) + 1, \qquad y = M*x, \tag{35}$$

$$f_{15}(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_D-1)^2\}$$
$$+\sum_{i=1}^{D} u(x_i, 10, 100, 4),$$
$$\text{where} \quad y_i = (1+\frac{1}{4}(x_i+1)), u(z_i,a,k,m) = \begin{cases} k(z_i-a)^m, & z_i > a, \\ 0, & -a \leq z_i \leq a, \\ k(-z_i-a)^m, & z_i < -a. \end{cases}$$
$$z = M*x \tag{36}$$

$$f_{16}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(y_i+0.5))] - D\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k * 0.5)]\right),$$
$$a=0.5, b=3, k_{\max}=20, y = M*x, \tag{37}$$

Composite Function 3 in [30] f_{17}: (Rotated CF3) is composed using ten Griewank functions, where each Griewank function is calculated with rotation.

Rotated Composite Function 3 in [31] f_{18}: f_{18} is composed using ten different benchmark functions: two rotated Rastrigin's functions, two rotated Weierstrass functions, two rotated Griewank's functions, two rotated Ackley's functions, and two sphere functions. f_{18} is more complex since the global optimum is not easy to locate.

TABLE I
BENCHMARK CONFIGURATIONS

Functions	Name	Dimension	Search Space	Threshold(ε)	Initial Range
$f_1(x)$	Sphere	30	$[-100, 100]^D$	0.01	$[-100, 50]^D$
$f_2(x)$	Rosenbrock	30	$[-10, 10]^D$	100	$[-10, 5]^D$
$f_3(x)$	Rastrigin	30	$[-5, 5]^D$	50	$[-5, 2]^D$
$f_4(x)$	Noncontinuous Rastrigin	30	$[-5, 5]^D$	50	$[-5, 2]^D$
$f_5(x)$	Ackley	30	$[-32, 32]$	0.01	$[-32, 16]^D$
$f_6(x)$	Griewank	30	$[-600, 600]^D$	0.01	$[-600, 200]^D$
$f_7(x)$	Penalized	30	$[-50, 50]^D$	0.01	$[-50, 20]^D$
$f_8(x)$	Weierstrass	30	$[-0.5, 0.5]^D$	0.01	$[-0.5, 0.2]^D$
$f_9(x)$	Rotated Quadric	30	$[-100, 100]^D$	100	$[-100, 50]^D$
$f_{10}(x)$	Rotated Rosenbrock	30	$[-10, 10]^D$	100	$[-10, 5]^D$
$f_{11}(x)$	Rotated Rastrigin	30	$[-5, 5]^D$	100	$[-5, 2]^D$
$f_{12}(x)$	Rotated Noncontinuous Rastrigin	30	$[-5, 5]^D$	100	$[-5, 2]^D$
$f_{13}(x)$	Rotated Ackley	30	$[-32, 32]^D$	0.01	$[-32, 16]^D$
$f_{14}(x)$	Rotated Griewank	30	$[-600, 600]^D$	100	$[-600, 200]^D$
$f_{15}(x)$	Rotated Penalized	30	$[-10, 10]^D$	0.01	$[-10, 5]^D$
$f_{16}(x)$	Rotated Weierstrass	30	$[-0.5, 0.5]^D$	10	$[-0.5, 0.2]^D$
$f_{17}(x)$	Rotated CF3	30	$[-500, 500]^D$	100	$[-5, 5]^D$
$f_{18}(x)$	Hybrid composite function	30	$[-500, 500]^D$	100	$[-5, 5]^D$

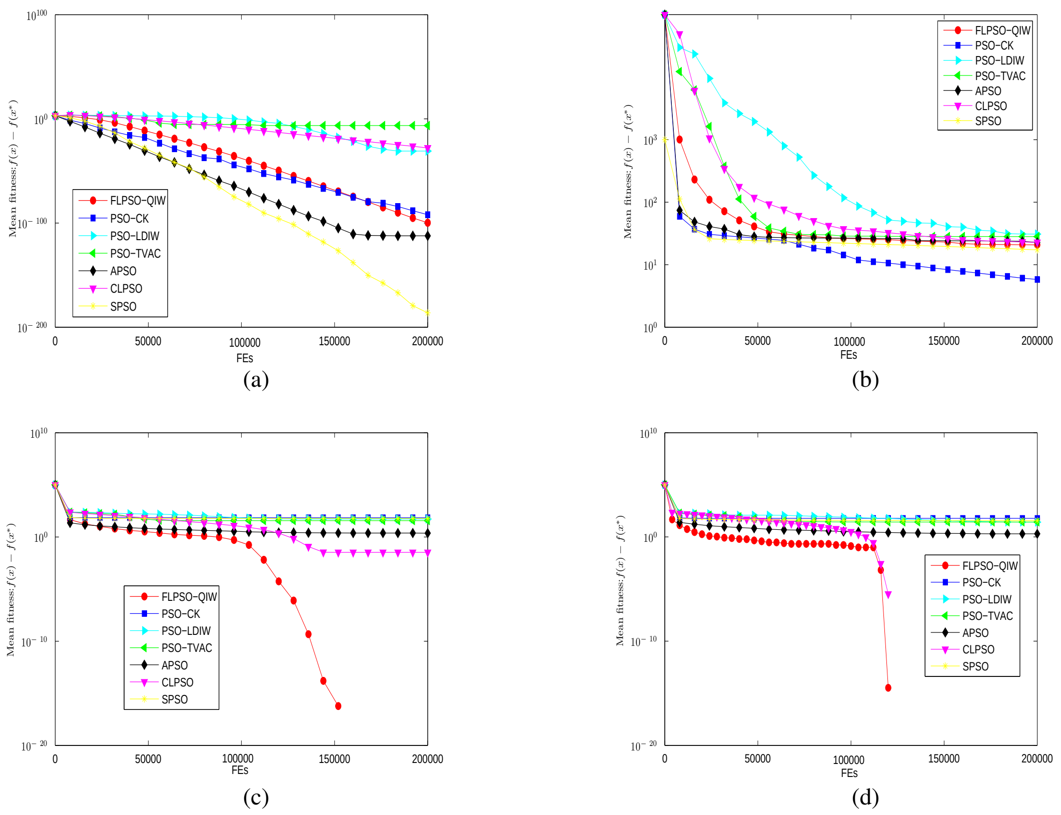

Fig. 1. Performance of the algorithms for four 30-dimensional benchmark functions.(a) f_1. (b) f_2. (c) f_3. (d) f_4.

The population size of all the PSOs is 20 except SPSO. For SPSO, as suggested in Ref. [24], the population size is 50. Six existing PSO algorithms are shown in Table II in detail. The first PSO is a standard PSO with local coupling topology and population size 50 [24]. PSO-LDIW [10, 14, 15] is with linearly decreasing inertia weight. PSO-TVAC [13] is a PSO with time-varying acceleration parameters and incorporating a self-organizing method. PSO-CK is a PSO with a constriction factor [9]. CLPSO delivers a comprehensive-learning strategy, which is used to yield better performance for multimodal functions [12].

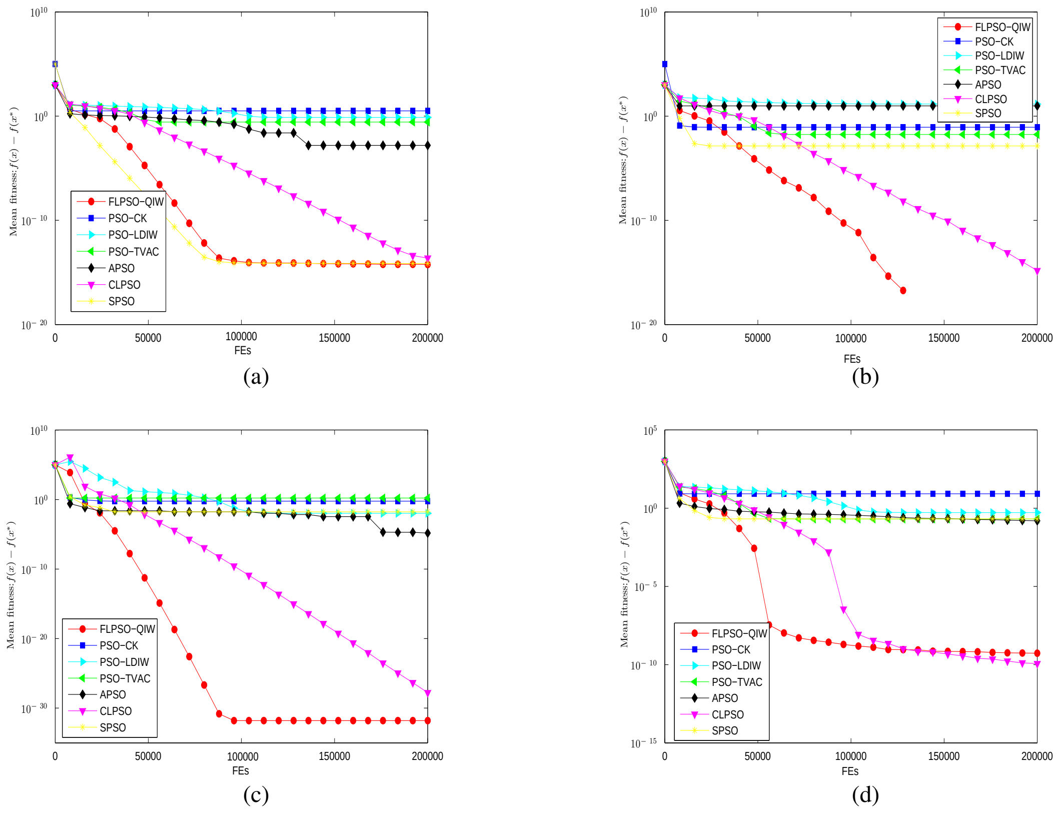

Fig. 2. Performance of the algorithms for four 30-dimensional benchmark functions.(a) f_5. (b) f_6. (c) f_7. (d) f_8.

APSO is an adaptive PSO, which can adjust the acceleration coefficients and inertia weight adaptively [16] according to an evolutionary factor by calculating average distance. The parameters for these PSOs are provided in Table II.

TABLE II
PSO ALGORITHMS FOR COMPARISON

Algorithm	Parameters	Reference
SPSO	$w : 0.729, c_1 : 1.49, c_2 : 1.49$,local structure	[24]
PSO-LDIW	$w : 0.9 - 0.4, c_1 = c_2 = 2$	[10, 14, 15]
PSO-TVAC	$w : 0.9 - 0.4, c_1 : 2.5 - 0.5, c_2 : 0.5 - 2.5$	[13]
PSO-CK	$w : 0.729, c_1 : 1.49, c_2 : 1.49$	[9]
CLPSO	$w : 0.9 - 0.4, c = 1.49, m = 7$	[12]
APSO	Automatically chosen	[16]
FLPSO-QIW	$w : 0.9 - 0.2, m = 1$	this paper

In all the experiments, the algorithm configuration of the FLPSO-QIW is listed as follows. The $\check{c}_1$ and $\check{c}_2$ are set to 1.5, which is close to the value of $c = 1.494$ in CLPSO. In (10)-(12), $\hat{c}_1$ and $\hat{c}_2$ are set to 2 and 1, respectively. In the following experiment, we will also show the bounds of $c_2 - c_1$ is vital for the search performance. κ_1 and κ_2 are set to 0.1 and 0.001, respectively.

All the algorithms use the same number of 2×10^5 fitness evaluations (FEs) for each test function. Further, all the experiments are performed on the same machine with a Core 2 2.26-GHz CPU, 2-GB memory, and Windows XP operating system. The initialization of each PSO is uniformly dispersed in Initial Range described in Table I. The results are obtained by following the similar methods in [12, 16, 31]. Each algorithm will repeat 30 times independently for eliminating random discrepancy.

TABLE III
SEARCH RESULT COMPARISONS AMONG SEVEN PSOS ON EIGHTEEN TEST FUNCTIONS AND COMPARISONS BETWEEN FLPSO-QIW AND OTHER PSOS ON T-TESTS

		SPSO	PSO-LDIW	PSO-TVAC	PSO-CK	CLPSO	APSO	FLPSO-QIW
f_1	Mean	**5.3e-189**	1.20e-31	4.14e-7	1.02e-92	1.12e-28	5.23e-113	1.09e-100
	Std. Dev.	**0**	6.04e-31	2.06e-6	5.08e-92	1.48e-28	**2.81e-112**	4.13e-103
	t-test	=	+	+	=	+	=	*
f_2	Mean	17.3	30.87	28.22	**5.81**	22.83	22.7	20.7
	Std. Dev.	2.3	24.26	14.22	**5.42**	14.9	19.2	19.1
	t-test	=	=	+	-	=	=	*
f_3	Mean	58.5	52.46	36.9	70.6	0.03	2.3	**0**
	Std. Dev.	11.7	42.38	8.67	21.9	0.18	7.1	**0**
	t-test	+	+	+	+	=	=	*
f_4	Mean	36.6	26.09	27.05	60.5	**0**	1.9	**0**
	Std. Dev.	12.0	17.27	9.39	17.1	**0**	6.0	**0**
	t-test	+	+	+	+	=	=	*
f_5	Mean	7.1e-15	0.75	0.26	3.1	2.27e-14	0.0015	**5.77e-15**
	Std. Dev.	1.1e-15	3.78	0.49	1.8	4.36e-15	0.0039	**1.80e-15**
	t-test	+	=	+	+	+	+	*
f_6	Mean	0.001	14.2	0.017	0.08	1.18e-15	9.0	**0**
	Std. Dev.	0.005	71.03	0.02	0.2	4.49e-15	52.3	**0**
	t-test	=	=	+	+	=	=	*
f_7	Mean	0.017	0.01	1.67	0.6	1.67e-28	1.5e-5	**1.5e-32**
	Std. Dev.	0.05	0.03	1.12e-15	0.92	1.66e-28	4.2e-5	**9.42e-34**
	t-test	+	+	+	+	+	+	*
f_8	Mean	0.2	0.52	0.2	8.4	**1.15e-10**	0.15	5.39e-10
	Std. Dev.	0.4	0.97	0.02	2.9	**1.2e-10**	0.1	4.28e-10
	t-test	+	+	+	+	=	+	*
f_9	Mean	6.5e-5	7579.07	4959.46	**9.9e-15**	725.55	9.89e-9	10.4
	Std. Dev.	0.0001	8281.14	6758.67	**3.4e-14**	375.08	1.4e-8	10.2
	t-test	-	+	+	-	+	-	*
f_{10}	Mean	35.3	45.59	51.49	**19.1**	28.7	29.0	32.3
	Std. Dev.	23	39.36	41.36	**17.8**	0.54	15.8	19.5
	t-test	=	=	+	-	=	=	*
f_{11}	Mean	69.9	81.12	61.76	78.9	88.0	84.0	**20.8**
	Std. Dev.	18	30.90	16.5	18.8	18.3	24.1	**17.2**
	t-test	+	+	+	+	+	+	*
f_{12}	Mean	71.6	85.68	63.03	88.9	84.3	82.6	**22.0**
	Std. Dev.	12.8	29.42	18.9	17.9	15.1	20.4	**16.2**
	t-test	+	+	+	+	+	+	*
f_{13}	Mean	1.3	3.56	2.77	3.0	0.0004	2.6	**6.6e-15**
	Std. Dev.	0.78	4.09	3.36	1.4	0.0009	0.9	**1.59e-15**
	t-test	+	+	+	+	+	+	*
f_{14}	Mean	0.005	0.011	0.02	0.04	2.4e-5	0.02	**3.32e-10**
	Std. Dev.	0.007	0.01	0.02	0.04	6.2e-5	0.03	**1.58e-9**
	t-test	+	+	+	+	=	+	*
f_{15}	Mean	0.29	0.14	1.67	1.8	0.11	0.02	**3.24e-20**
	Std. Dev.	0.54	0.34	1.12e-15	2.9	0.14	0.03	**1.52e-19**
	t-test	+	+	+	+	+	+	*
f_{16}	Mean	13.1	7.81	8.4	15.5	10.67	14.7	**0.1**
	Std. Dev.	3.3	2.50	2.5	3.4	1.62	3.3	**0.2**
	t-test	+	+	+	+	+	+	*
f_{17}	Mean	1265.99	1883	1883	1866	22.9	709.3	**7.7**
	Std. Dev.	437	154	154	227	47.4	504	**27.3**
	t-test	+	+	+	+	=	+	*
f_{18}	Mean	274	360	248	236	162	332	**76.2**
	Std. Dev.	164	197	214	84	8.5	143	**13.6**
	t-test	+	+	+	+	+	+	*
	Score	12	14	18	12	10	10	*

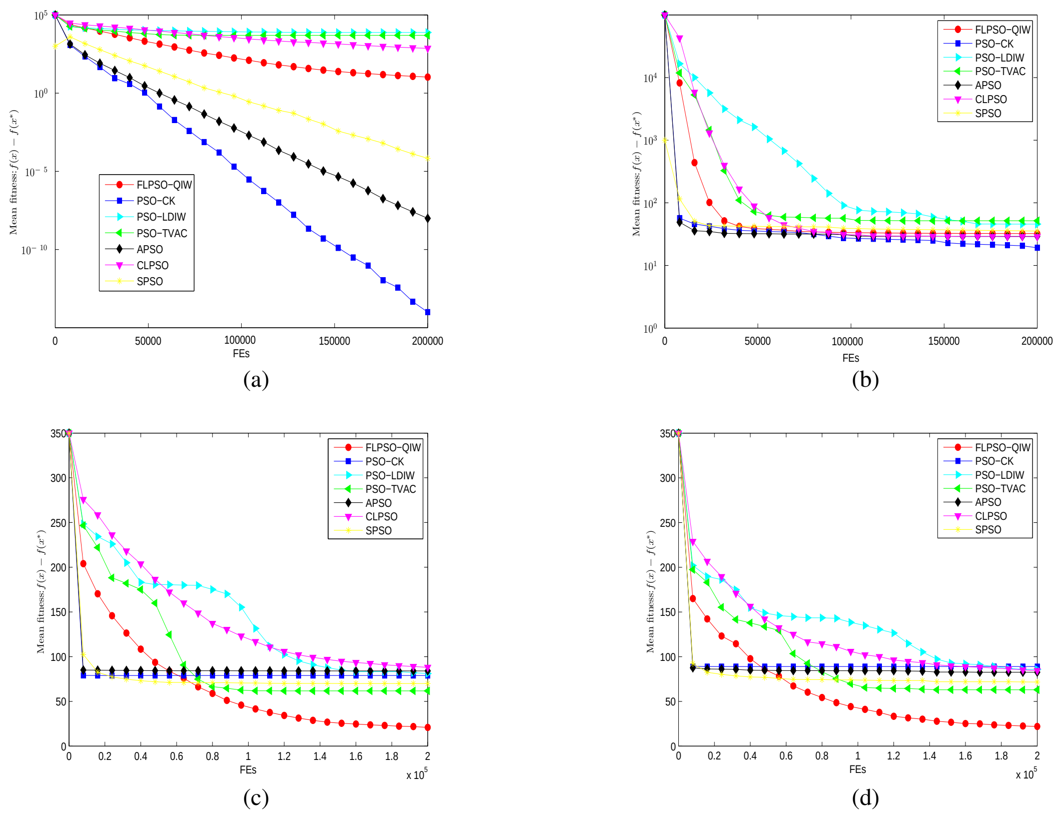

Fig. 3. Performance of the algorithms for four 30-dimensional benchmark functions.(a) f_9. (b) f_{10}. (c) f_{11}. (d) f_{12}.

B. Comparisons on the solution accuracy

The mean solutions and standard deviation (Std. Dev.) of the solutions in 30 independent runs are listed in Table III. The best result among those PSOs is indicated by Boldface in the table. Figs. 1-5 show the comparisons in terms of convergence, mean solutions and evolution processes in solving 18 benchmark functions.

From the Table III and Figs. 1-5, the FLPSO-QIW provides the best performance on the $f_3 - f_7$, $f_{11} - f_{18}$ and ranks the second on f_1, f_2 and f_8. SPSO can reach the highest accuracy on f_1. FLPSO-QIW delivers good performance on f_1, which is used to test the convergent rate. PSO-CK offers the best performance on f_2, f_9 and f_{10}. The results show that SPSO and PSO-CK have good ability of convergence speed. On multimodal optimization problems without rotation, FLPSO-QIW can achieve the highest performance on $f_3 - f_8$ among seven PSOs, as seen from Table III, Figs. 1(c)-(d) and Fig. 2. CLPSO ranks second on these benchmarks f_3, f_4, f_6, f_7 on accuracy and performs a little better than FLPSO-QIW on f_8 with a slow convergence speed. On rotated multimodal problems $f_{11} - f_{16}$, FLPSO-QIW performs much better than other PSOs. Table III and Figs. 3(c)-(d) and Fig. 4 illustrate the competitive performance of FLPSO-QIW. One can observe that the proposed method can help the PSO to search the optimum as well as maintaining a higher convergence speed when dealing with multimodal rotated functions. On composite functions f_{17} and f_{18}, FLPSO-QIW can deliver best search performance and high convergence speed among these PSOs, as illustrated in Table III and Fig. 5. The capabilities of avoiding local optima and finding global optimum of multimodal functions indicate the superiority of FLPSO-QIW.

For a thorough comparison, the t-test has also been carried out in this paper. Table III presents the score on every function of this two-tailed test with a significance level of 0.05 between the FLPSO-QIW and another PSO algorithm. Rows "+ (Better)," "= (Same)," and "- (Worse)" give the number of functions that the FLPSO-QIW performs significantly better than, almost the same as, and significantly worse than the compared algorithm on fitness values in 30 runs, respectively. Row "Score" shows the difference between

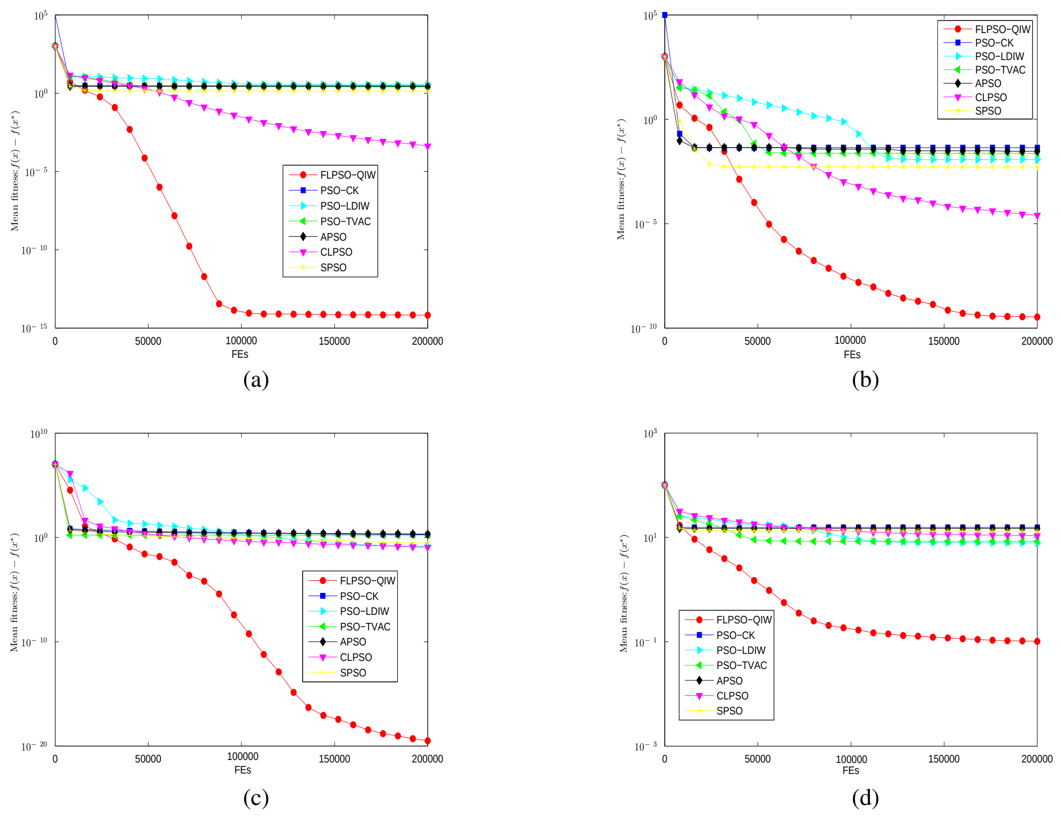

Fig. 4. Performance of the algorithms for four 30-dimensional benchmark functions.(a) f_{13}. (b) f_{14}. (c) f_{15}. (d) f_{16}.

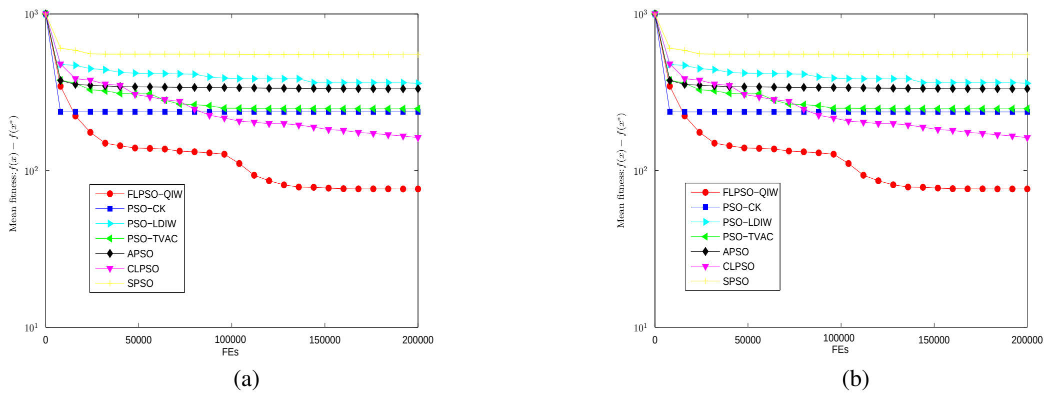

Fig. 5. Performance of the algorithms for four 30-dimensional benchmark functions.(a) f_{17}. (b) f_{18}.

the number of +'s and the number of −'s, which is used to give an overall comparison between the two algorithms. For instance, comparing the FLPSO-QIW and the SPSO, the former significantly outperforms the latter on seven functions (f_3, f_4, f_5, f_7, f_8, f_{11}, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17} and f_{18}), does almost the same as the latter on four functions (f_1, f_2, f_6 and f_{10}), and does worse on f_9 function, yielding a Score merit of $13-1=12$, indicating that the FLPSO-QIW generally outperforms the SPSO. Although it worked slightly weaker on some functions, the FLPSO-QIW in general offered much improved performance than all the PSOs compared, as confirmed in Table III. It is also worth pointing out that the FLPSO-QIW performs much better than the other PSOs on rotated and composite optimization problems.

Comparing the results and the convergence graphs, among these seven PSO algorithms, the PSO-LDIW and the PSO-TVAC cannot perform well on all the functions. The CLPSO has good global search ability and slow convergence speed. The APSO can converge to the best solution found so far very quickly

TABLE IV
CONVERGENCE SPEED AND ALGORITHM RELIABILITY COMPARISONS; '-' REPRESENTING NO RUNS REACHED AN ACCEPTABLE SOLUTION

		SPSO	PSO-LDIW	PSO-TVAC	PSO-CK	CLPSO	APSO	FLPSO-QIW
f_1	Mean FEs	14968	105803	45669	8953	56172.2	**7176**	26730.2
	Right Percentage(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_2	Mean FEs	8373.72	97634	40850.3	**5191**	54753.4	6099	24911.2
	Right Percentage(%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**
f_3	Mean FEs	**2511.8**	100387	47345.1	3225	54847.3	3474	6899.63
	Right Percentage(%)	23.3	50	90	13.3	**100**	**100**	**100**
f_4	Mean FEs	25385.6	120030	51313.5	13334	43254.6	3341	**3180**
	Right Percentage(%)	83.3	90	**100**	36.7	**100**	**100**	**100**
f_5	Mean FEs	**18659.8**	111933	50494.8	-	63097.7	56178	35045.4
	Right Percentage(%)	**100**	96.7	76.7	0	**100**	96.7	**100**
f_6	Mean FEs	10490.1	111273	48358.6	**8878**	64260.7	10543	33022.2
	Right Percentage(%)	90	43.3	50	23.3	**100**	56.7	**100**
f_7	Mean FEs	16990	97955	-	**15091**	46810	33399	23155
	Right Percentage(%)	86.7	90	0	43.3	**100**	**100**	**100**
f_8	Mean FEs	**16260.4**	118947	-	-	78017.4	-	44116.6
	Right Percentage(%)	56.7	60	0	0	**100**	0	**100**
f_9	Mean FEs	40293.3	129523	63162	**18811**	-	21808	106032
	Right Percentage(%)	**100**	33.3	53.3	**100**	0	**100**	**100**
f_{10}	Mean FEs	8517.48	95260.2	39344.5	**5089**	46849.5	5297	25981.3
	Right Percentage(%)	**100**	90	80	**100**	**100**	**100**	**100**
f_{11}	Mean FEs	11165.7	100387	61298.9	**2767**	121104	4199.80	50162.8
	Right Percentage(%)	96.7	50	96.7	90	70	80	**100**
f_{12}	Mean FEs	11695.8	123628	68345.9	**2606**	111435	18110	45624.4
	Right Percentage(%)	93.3	70	96.7	73.3	83.3	86.7	**100**
f_{13}	Mean FEs	**4948.72**	-	49442	-	112375	-	37610.3
	Right Percentage(%)	20	0	3.3	0	96.7	0	**100**
f_{14}	Mean FEs	505.28	113561	50322.7	13613	73009	10510	32733.3
	Right Percentage(%)	**100**	53.3	36.7	20	**100**	43.3	**100**
f_{15}	Mean FEs	77905	131640	-	86552	122008	75667	**39062**
	Right Percentage(%)	40	56.7	0	26.7	56.7	23.3	**100**
f_{16}	Mean FEs	4361.2	96677.7	41340.7	-	125887	-	**15443.3**
	Right Percentage(%)	13.3	86.7	70	0	23.3	0	**100**
f_{17}	Mean FEs	-	-	-	-	75300.8	76890	**33402**
	Right Percentage(%)	0	0	0	0	90	10	**93.3**
f_{18}	Mean FEs	-	-	15211	-	-	-	**142270**
	Right Percentage(%)	0	0	33.3	0	0	0	**100**
	Mean Reliability	66.8	59.4	54.8	40.3	78.9	55.3	**99.6**

though it is easy to stuck in the local optima. The SPSO and the PSO-CK have good search capability on unimodal optimization problems. The FLPSO-QIW has good local search ability and global search ability at the same time.

C. Comparisons on convergent rate and successful percentage

The convergent rate for achieving the global optimum is another key point for testing the algorithm performance. Note that in solving real-world optimization problems, the "FE" overwhelms the algorithm overhead. Hence, the computation times of these algorithms are not provided here. Table IV shows that FLPSO-QIW needs least FEs to achieve the acceptable solution on f_4, f_{15}, f_{16}, f_{17} and f_{18}, which reveals that FLPSO-QIW has a higher convergent rate than other algorithms. ALthough SPSO, PSO-CK or APSO might outperform FLPSO-QIW on the other functions, SPSO, PSO-CK and APSO have much worse successful ratio and accuracy than FLPSO-QIW on the tested functions. In addition, FLPSO-QIW can achieve accepted value with a good convergence speed and accuracy on most of the functions, as seen from Figs. 1-5 and Table IV. Table IV also shows that FLPSO-QIW yields a highest percentage for achieving acceptable solutions in 30 runs. According to the no free lunch theorem [33], any elevated performance over one class of problems is offset by performance over another class. Hence, one algorithm cannot perform better on convergence speed and accuracy than the others on every optimization problem.

In summary, the FLPSO-QIW performs best on multimodal functions with or without rotation and has good search ability of unimodal function. Owing to the proposed techniques, the FLPSO-QIW processes capabilities of fast convergence speed, the highest successful ratio and the best search accuracy among these PSOs.

D. Analysis on the Computational Complexity of Algorithms

Following the methods in [31], the computational complexity of the algorithms discussed in this paper is computed as in Table V. Computations of T_0, T_1, and $\hat{T}_2$ can be referred in [31] and thus one can obtain $(\hat{T}_2 - T_1)/T_0$. The results of PSO-LDIW, PSO-TVAC, SPSO, PSO-CK, APSO, CLPSO and FLPSO-QIW are obtained in Table V. All of the time values are measured in CPU seconds. The results of PSO-LDIW, PSO-TVAC, SPSO, PSO-CK, APSO, CLPSO and FLPSO-QIW are obtained on a PC with CPU Dual Core 2.26 GHz, RAM 2G, Platform: MATLAB 2009a. The computational complexity of FLPSO-QIW is modest, with a complexity value near 2.

TABLE V
COMPUTATIONAL COMPLEXITY OF ALGORITHMS

Algorithm	T_0	T_1	$\hat{T}_2$	$(\hat{T}_2 - T_1)/T_0$
SPSO	1.95	18.72	41.59	11.72
PSO-LDIW	1.95	18.72	41.45	11.65
PSO-TVAC	1.95	18.72	41.83	11.85
PSO-CK	1.95	18.72	40.80	11.33
CLPSO	1.95	18.72	20.53	0.92
APSO	1.95	18.72	40.65	11.24
FLPSO-QIW	1.95	18.72	22.89	2.13

E. Performance of feedback mechanism and ELS

Fitness feedback mechanism and ELS are also used to test them on the search performance of FLPSO-QIW. The fitness feedback mechanism is the method described in Eqs. (9)-(12). The ELS method is illustrated in Eqs. (15)-(21). We compare the effectiveness of FLPSO-QIW without feedback mechanism, FLPSO-QIW without ELS and complete FLPSO-QIW here to show the performance of feedback mechanism and ELS. Results of 30 independent runs and comparisons between FLPSO-QIW with other variants of FLPSO-QIW on t-test of fitness value are shown in Table VI. The explanation of symbols "$+$", "$-$" and "$=$" are provided in Section III-B.

From the results, it can be seen that FLPSO-QIW with both ELS and feedback mechanism is the best among three variants of FLPSO. FLPSO-QIW with two techniques can search with highest accuracy and fastest convergence speed. For t-test of fitness value, it can be found that FLPSO-QIW performs significantly better on f_4 than FLPSO-QIW without ELS. It can also be observed that FLPSO-QIW performs significantly better on f_5 than FLPSO-QIW without feedback mechanism. FLPSO-QIW can not only deliver a highest accuracy on unimodal functions, but also offer a good global search performance on multimodal functions.

To summarize, the full FLPSO-QIW is the most powerful for the tested functions. The results verify that fitness feedback method can accelerate the convergence and enhance accuracy and ELS can help the swarm to have a better search performance.

F. Performance of quadratic inertia weight and its parameters setting

In this subsection, the quadratic inertia weight with linear inertia weight is compared. Moreover, the parameter setting is also considered. In all the experiments, w_1 is set 0.9 so that only w_2 is adjusted in

TABLE VI

ADVANTAGES OF FEEDBACK MECHANISM AND ELS

Algorithms	Functions	f_1	f_4	f_5	f_6	f_{11}	f_{13}
FLPSO-QIW without ELS	Average	9.69e-97	0.56	6.36e-15	0	37	5.65e-15
	Std. Dev.	4.29e-96	0.81	1.7e-15	0	6	1.8e-15
	FEs	26625	48926	31455	30060	71136	34979
	t-test	=	+	=	=	=	=
FLPSO-QIW without feedback approach	Average	5.7e-46	0	8.3e-15	4.04e-14	33.4	7.9e-15
	Std. Dev.	2.2e-45	0	2.4e-15	1.5e-13	19.5	1.4e-15
	FEs	36175	3475	41477	51335	72681.9	46821
	t-test	=	=	+	=	=	=
FLPSO-QIW	Average	1.09e-100	0	5.77e-15	0	20.8	6.6e-15
	Std. Dev.	4.13e-103	0	1.8e-15	0	17.2	1.6e-15
	FEs	26730	3180	35045	33022	50162	37610

TABLE VII

PERFORMANCE OF QUADRATIC INERTIA WEIGHT AND ITS PARAMETERS SETTING(THE BEST VALUE IS IN BOLD FONT.)

	Functions	f_1	f_4	f_5	f_6	f_{11}	f_{13}
linear inertia weight ($w_2 = 0.4$)	Average	1.1E-71	**0**	7.43e-15	4.81e-17	35.7	7.43e-15
	FEs	40302	3321	59796	56731	87909	63360
	t-test	=	=	+	=	+	=
linear inertia weight ($w_2 = 0.2$)	Average	1.04e-85	**0**	6.72e-15	**0**	22.2	6.37e-15
	FEs	34768	3309	49075	47100	65277	52038
	t-test	=	=	=	=	=	=
quadratic inertia weight ($w_2 = 0.3$)	Average	3.8e-96	**0**	6.48e-15	2.47e-4	31.2	6.37e-15
	FEs	28322	3140	37839	35290	72472	40503
	t-test	=	=	=	=	=	=
quadratic inertia weight ($w_2 = 0.1$)	Average	4.7e-100	0.03	5.18e-15	**0**	24.9	**6.3e-15**
	FEs	**24907**	**3247**	**32912**	**31943**	**49480**	**35391**
	t-test	=	=	=	=	=	=
quadratic inertia weight ($w_2 = 0.2$)	Average	**1.09e-100**	**0**	5.77e-15	**0**	**20.8**	6.6e-15
	FEs	26730	3180	35045	33022	50162	37610

this paper. Results of 30 independent runs and comparisons between FLPSO-QIW with other parameter setting of FLPSO-QIW on t-test of fitness value are illustrated in Table VII.

The results on $f_1, f_4, f_5, f_6, f_{11}, f_{13}$ are listed in Table VII. It can be observed that PSO with quadratic inertia weight can achieve faster convergence speed than that with linear inertia weight. In addition, a smaller w_2 will lead to a higher convergence speed than a larger one, which is consistent with the description in [14]. However, PSO with a small w_2 will get into the local optima easier than that with a large one when testing f_4. The results of t-test only show that FLPSO-QIW significantly performed better than FLPSO with linearly decreased inertia weight $w_2 = 0.4$. The comparisons in t-test between FLPSO-QIW with other three variants of FLPSO-QIW show that FLPSO-QIW performs almost the same as other three variants of FLPSO-QIW, which indicates that FLPSO-QIW is not sensitive to the adjustment of parameters. In order to make a balance of high accuracy and rapid search speed, $w_2 = 0.2$ is selected in our paper. It is worth noting that other parameters can be chosen according to the different requirements of application.

G. Performance of time-varying acceleration parameters

In this subsection, the effects of $\hat{c}_1$ and $\hat{c}_2$ are investigated on FLPSO-QIW here. For the sake of simplicity, in this paper, the mean values of $\hat{c}_1$ and $\hat{c}_2$ are both set to 1.5, which is close to the value 1.494 adopted in CLPSO. After fixing the mean value, the range of $\hat{c}_1 - \hat{c}_2$ is tuned gradually. It can be observed from Table VIII that a larger range of $\hat{c}_1 - \hat{c}_2$ can lead to a faster convergence speed of PSO, as seen from Table VIII. The comparisons in t-test between FLPSO-QIW with other three parameter settings of FLPSO-QIW show that FLPSO-QIW perform almost the same as other three parameter settings of FLPSO-QIW, which indicates that FLPSO-QIW is not sensitive to adjustment of parameters. In order to make a balance of rapid convergence rate and search accuracy, $\hat{c}_1 = 2$ and $\hat{c}_2 = 1$ are used as a

TABLE VIII
EFFECTS OF $\hat{c}_1$ AND $\hat{c}_2$ ON SEARCH ACCURACY AND CONVERGENCE RATE(THE BEST VALUE IS IN BOLD FONT.)

		$\hat{c}_1 = 2, \hat{c}_2 = 1$	$\hat{c}_1 = 2.25, \hat{c}_2 = 0.75$	$\hat{c}_1 = 1.75, \hat{c}_2 = 1.25$	$\hat{c}_1 = \hat{c}_2 = 1.5$
f_1	Average	1.09e-100	**1.1e-102**	9.1e-99	7.6e-49
	Mean FEs	26730	**24144**	28723	31312
	t-test	*	=	=	=
f_4	Average	**0**	**0**	**0**	**0**
	Mean FEs	**3180**	3256	3240	3224
	t-test	*	=	=	=
f_5	Average	5.77e-15	**5.65e-15**	5.77e-15	6.3e-15
	Mean FEs	35045	36038	35130	**35044**
	t-test	*	=	=	=
f_6	Average	**0**	**0**	**0**	**0**
	Mean FEs	33022	28996	39737	42640
	t-test	*	=	=	=
f_{11}	Average	**20.8**	24	29.8	22.3
	Mean FEs	**50162**	51970	52767	51402
	t-test	*	=	+	=
f_{14}	Average	3.3e-10	0.0002	6.36e-12	**3.6e-14**
	Mean FEs	33402	**30652**	40756	42770
	t-test	*	=	=	=

representative parameter setting in our paper.

IV. CONCLUSION

In this paper, a feedback learning PSO with quadratic inertia weight has been proposed to solve various types of optimization problems. In the proposed PSO, the fitness information is used to control the parameters of PSO. The acceleration coefficients are determined by both generation time and search environment. The inertia weight is calculated by a quadratic function. A large number of experiments verify that the feedback learning strategy enables the FLPSO-QIW to make use of the information in swarm. In comparison with six PSO variants, the FLPSO-QIW works in a more effective way and find better quality solutions more frequently.

V. ACKNOWLEDGEMENTS

The authors are grateful to the Editor-in-Chief, Associate Editor and anonymous reviewers for their careful reading and constructive comments.

REFERENCES

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference On Neural Network, 1995, pp. 1942-1948.

[2] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga MS, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 256-279, Jun. 2004.

[3] R. A. Krohling and L.dos Santos Coello, Coevolutionary particle swarm optmization using Gaussian distribution for solving constrned optimization problems, IEEE Trans. Syst., Man, Cybern. B, vol.36, no.6, pp.1407-1416, Dec. 2006.

[4] X. Cai, Z. Cui, J. Zeng, and Y. Tana. Dispersed particle swarm optimization. Information Processing Letters, 105:231-235, 2008.

[5] P. S. Andrews, An investigation into mutation operators for particle swarm optimization, in Proc. IEEE Congr. Evol. Comput., Vancouver, BC, Canada, 2006, pp. 1044-1051.

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

[7] F. VD. Bergh and A. P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. , vol. 8.,no. 3 pp.225-239, June 2004.

[8] Y. P. Chen, W. C. Peng, and M. C. Jian, Particle swarm optimization with recombination and dynamic linkage discovery, IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 37, no. 6, pp. 1460-1470, Dec. 2007.

[9] M. Clerc, J. Kennedy, The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space, IEEE Transactions on Evolutionary Computation 6(1). Piscataway, NJ, 2002, pp. 58-73.

[10] R. C. Eberhart and Y. H. Shi, Particle swarm optimization: Developments, applications and resources, in Proc. IEEE Congr. Evol. Comput. Seoul, Korea, 2001, pp. 81-86.

[11] 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.

[12] J.J. Liang, 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.

[13] A. Ratnaweera, SK. Halgamure, HC. Watson. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol. Comput. 2004;8:240-55.

[14] Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, in Proc. IEEE Congr. Evol. Comput., 1998, pp. 69-73.

[15] Y. Shi, RC Eberhart. Parameter selection in particle swarm optimization. In: Proceedings of the 7th international conference on evolutionary programming VII. LNCS, vol. 1447. New York: Springer-Verlag; 1998. p. 591-600.

[16] Z. Zhan, J. Zhang, Y. Li, H.S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. System, man and cybernetics-B, 39(2009)1362-1381.

[17] J. Kennedy, The particle swarm social adaptation of knowledge, in Proc. IEEE Int. Conf. Evol. Comput., Indianapolis, IN, Apr. 1997, pp. 303-308.

[18] J. Kennedy, Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. Proceedings of the 1999 Conference on Evolutionary Computation, 1931-1938.

[19] J. Kennedy and R. Mendes, Population structure and particle swarm performance, in Proc. IEEE Congr. Evol. Comput., Honolulu, HI, 2002, pp. 1671-1676.

[20] J. Kennedy and R. Mendes, Neighborhood topologies in fully informed and best-of-neighborhood particle swarms, IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 36, no. 4, pp. 515-519, Jul. 2006.

[21] S. Hsieh, T. Sun, C. Liu, and S. Tsai, Efficient Population Utilization Strategy for Particle Swarm Optimizer, IEEE Trans. On SMC-B, VOL. 39, NO. 2, pp. 444-456, 2009.

[22] M. Lovbjerg, T. K. Rasmussen, and T. Krink, Hybrid particle swarm optimizer with breeding and subpopulations, in Proc. Genetic Evol. Comput. Conf., 2001, pp. 469-476.

[23] K. Deb and N. Padhye, Improving a Particle Swarm Optimization Algorithm Using an Evolutionary Algorithm Framework, KanGAL Report Number 2010003.

[24] D. Bratton; J. Kennedy. Defining a standard for particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium. Honolulu, United States: 2007.

[25] B. Liu, L. Wang, Y. H. Jin, F. Tang, and D. X. Huang, Improved particle swarm optimization combined with chaos, Chaos, Solitons Fractals, 25(5), 1261-1271, 2005.

[26] W. Hong, Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model, Energy Conversion and Management, 50(1), 105-117, 2009.

[27] Z. Wang, D. W. C. Ho., and X. Liu. Variance-constrained filtering for uncertain stochastic systems with missing measurements. IEEE Transactions on Automatic Control, 48(2003)1254-1258.

[28] Z. Wang, F. Yang, Daniel W. C. Ho, and X. Liu, Robust H_∞ Control for Networked Systems With Random Packet Losses, IEEE Trans Systems, man and cybernetics-PART B, VOL. 37, NO. 4, AUGUST 2007.

[29] X. Yao, Y. Liu and G. M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput., vol.3, no.2, pp.82-102, July,. 1999.

[30] J. J. Liang, P. N. Suganthan, and K. Deb, Novel composition test functions for numerical global optimization, in Proc. Swarm Intell. Symp., Jun. 2005. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan.

[31] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, in Proc. IEEE Congr. Evol. Comput.,2005, pp.1-50.

[32] R. Salomon, Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions, BioSystems, vol. 39, pp. 263-278, 1996.

[33] D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, IEEE Congr. Evol. Comput. vol. 1, no, 1, pp. 67-82, Apr. 1997.