Information Sciences 221 (2013) 355–370

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A new fitness estimation strategy for particle swarm optimization

Chaoli Sun [a,*], Jianchao Zeng [a], Jengshyang Pan [b,c], Songdong Xue [a], Yaochu Jin [d]

[a] Complex System and Computational Intelligence Laboratory, Taiyuan University of Science and Technology, Taiyuan, Shanxi 030024, China
[b] Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, Guangdong 518055, China
[c] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 807, Taiwan
[d] Department of Computing, University of Surrey, Guildford GU2 7XH, United Kingdom

ARTICLE INFO

Article history:
Received 22 June 2011
Received in revised form 5 September 2012
Accepted 20 September 2012
Available online 29 September 2012

Keywords:
Particle swarm optimization
Fitness evaluation
Fitness estimation
Computationally-expensive optimization problem

ABSTRACT

Particle swarm optimization (PSO) is a global metaheuristic that has been proved to be very powerful for optimizing a wide range of problems. However, PSO requires a large number of fitness evaluations to find acceptable (optimal or sub-optimal) solutions. If one single evaluation of the objective function is computationally expensive, the computational cost for the whole optimization run will become prohibitive. FESPSO, a new fitness estimation strategy, is proposed for particle swarm optimization to reduce the number of fitness evaluations, thereby reducing the computational cost. Different from most existing approaches which either construct an approximate model using data or utilize the idea of fitness inheritance, FESPSO estimates the fitness of a particle based on its positional relationship with other particles. More precisely, Once the fitness of a particle is known, either estimated or evaluated using the original objective function, the fitness of its closest neighboring particle will be estimated by the proposed estimation formula. If the fitness of its closest neighboring particle has not been evaluated using the original objective function, the minimum of all estimated fitness values on this position will be adopted. In case of more than one particle is located at the same position, the fitness of only one of them needs to be evaluated or estimated. The performance of the proposed algorithm is examined on eight benchmark problems, and the experimental results show that the proposed algorithm is easy to implement, effective and highly competitive.

1. Introduction

Many real-world engineering problems are computationally expensive or black-box instances. A black-box function is an unknown function which, given a set of inputs, produces a corresponding set of outputs without knowing its exact mathematical description or internal structure [46]. Typically, the outputs of the black-box systems are obtained by performing finite element analysis or computational fluid dynamics simulations, which are usually computationally expensive. For example, vehicle crashworthiness optimization problems normally require a large number of crash simulations to achieve the optimal design [53], which is a widely known computationally expensive process [24]. The classical approach to reduce the number of expensive function evaluations and thereby to reduce the overall computational time of the optimization procedure is to model the objective function by a so-called meta-model [56], sometimes also known as surrogate or fitness estimation. In this paper, the term fitness estimation is adopted, which evaluates the quality of some candidate solutions using a

* Corresponding author. Tel.: +86 3516998016.
E-mail addresses: clsun1225@163.com (C. Sun), zengjianchao@263.net (J. Zeng), jspan@cc.kuas.edu.tw (J. Pan), xuesongdong@163.com (S. Xue), yaochu.jin@surrey.ac.uk (Y. Jin).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.09.030

meta-model instead of the real expensive function. Generally, fitness estimation techniques can be categorized into three main approaches [28]. The first approach, widely employed in evolutionary algorithms, is termed fitness inheritance, in which the fitness of an offspring individual is inherited from its parents. The second type, also the most common one, is to construct a model, such as neural networks [17,55], response surface methodology [32], kriging[23], support vector machines [34] and radial basis function network [41] using a set of training data which are usually sampled directly by the optimizer. The accuracy of the constructed model depends on the sampled data, which heavily influences the quality of the found optimal solution. In the last and recently proposed approach, the search is guided by a meta-model and a data sampling technique such as the mode-pursuing sampling approach [54], which is an extension of the random-discretization based sampling method suggested in [18].

Evolutionary Algorithms (EAs), which have been successfully applied in various fields of science and technology [8,40], such as multidisciplinary rotor blade design [19] and aircraft wing design [42], have been proved to be powerful global optimizers. Generally, EAs outperform conventional optimization algorithms in finding the global optima of multi-modal problems which are discontinuous, non-differentiable, noisy and multi-criteria. However, most EAs require a large number of fitness evaluations, even for solving a simple 2-dimensional design problem, before an acceptable global or local optimum is found [21,54]. This makes the evolutionary optimization process highly time-consuming, especially for computationally expensive black-box problems. For example, the computational time required for a Genetic Algorithm (GA) based optimization routine takes about one month on an average performance PC, where a viscous flow solver is called for about 2000 times during the aerodynamic optimization process [45]. Therefore, most EAs, though capable of finding a global optimal solution, are inefficient for optimizing computationally expensive problems. Recently, with the emergence of new estimation techniques, surrogate-assisted EAs have received considerably increasing interest [8]. Buche et al. [1] incorporated a Gaussian model as an inexpensive fitness estimator in EAs. An improved kriging-assisted multi-objective GA was proposed by Li [28] for multi-objective optimization of real-world engineering problems, in which the kriging method is adopted to assist a traditional multi-objective genetic algorithm. Bouzarkouna et al. [3] coupled a local quadratic model with the covariance matrix adaptation evolution strategy (CMA-ES) so as to reduce the number of evaluations using the real objective function.

Particle swarm optimization (PSO) is a global metaheuristic proposed by Kennedy and Eberhart in [13,25]. The idea of PSO was based on simplified models simulating some social behaviors of animals, such as bird flocking and fish schooling. Because of its simplicity in concept, fast convergence, and strong global optimization capability, PSO has received increasing attention and has been successfully applied to solving a large class of unconstrained optimization problems [4,12,60], such as reactive power optimization [9,30,31], neural network training [15,36] and data mining [29,47,59]. Like most EAs, PSO does not require that the objective functions be continuous or differentiable [26]. In addition, the search dynamics of the PSO is described by an explicit updating formula that is easy to analyze. However, in the real world, optimization problems are often quite complex, where a number of constraints must be satisfied, more than one objective function needs to be minimized; some fitness functions cannot be explicitly expressed (such as black-box problems), and fitness evaluation is computationally expensive. Many approaches have been proposed to solve constrained and multi-objective optimization problems based on PSO [2,35,50–52,57,58]. However, not much attention has been paid to problems which have no explicit model for fitness evaluations or involve noisy and extremely time-consuming fitness evaluation. Just like most other EAs, PSO also requires a large number of fitness evaluations before an acceptable global or local optimal solution is found. PSO-based optimization algorithms assisted by meta-models, although not as many as GA, have been proposed for solving the computationally expensive problems. Reyes-Sierra and Coello [43] applied fitness inheritance and approximation techniques in the PSO-based multi-objective algorithm presented in [49] to reduce the number of fitness evaluations. Hendtlass [20] proposed to estimate the fitness by adding reliability on each particle's fitness value on top of the approach used in [44], where the fitness of a new individual is estimated based on the fitness of its parents and the distance between this new individual and its parents in the genotype space. Cui et al. [11] also introduced the fitness inheritance method into PSO to reduce the number of fitness evaluations, which is called a "fast particle swarm optimization" (FPSO). To develop a more effective algorithm for solving stochastic problems, Cui et al. [10] suggested an intelligent algorithm that combines a generalized regression neural network for fitness estimation with PSO. In general, most surrogate-assisted PSO algorithms either use fitness inheritance, or build meta-models based on samples composed of individuals in one or more generations. The former often needs additional memory space to store fitness values of the parents, whereas the latter requires highly accurate surrogates in order to obtain the global optima. In this paper, a simpler and more effective fitness estimation approach is presented. Different from the above-mentioned approaches, the fitness of an individual is estimated by that of any other individuals based on the positional relationship of these two individuals. The positional relationship between any two individuals is obtained by the updating formula of the PSO. The main advantage of the proposed approach is that it can estimate the fitness of individuals using the updating formula in PSO without the need to construct meta-models that demands additional sampling of the expensive objective function, thereby significantly reducing the computational cost.

The remainder of this paper is organized as following. Section 2 gives a brief description of the canonical PSO. In Section 3, a new fitness estimation strategy for PSO is proposed. Section 4 presents experimental results and analysis. Finally, conclusions and suggested future work are provided in Section 5.

2. Particle Swarm Optimization (PSO)

In a PSO, it is assumed that a swarm of particles, each of which has no volume and no weight representing a candidate solution of a given optimization problem, move in a D-dimensional search space $S \subset R^D$ with a certain velocity. The movement dynamics of each particle in the search space is governed by its current position and velocity,where the current velocity is determined by its previous velocity and its distance to the position where this particle has achieved its best fitness so far (personal best) and to the position of the particle that has achieved the best fitness among all particles (global best).

For a particle i, its position can be described by a D-dimensional vector $\vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \in S$ and its velocity is also a D-dimensional vector $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \in S$. The position at which the particle i has achieved its personal best fitness, called p_{best}, is a point in S, which is denoted as $\vec{p}_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. Similarly, the position at which the global best fitness has achieved, called g_{best}, is denoted as $\vec{p}_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$. For particle i, the velocity and position are updated as follows [14]:

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1\mathbf{r}_1(\vec{p}_i(t) - \vec{x}_i(t)) + c_2\mathbf{r}_2(\vec{p}_g(t) - \vec{x}_i(t)) \tag{1}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{2}$$

where ω is a parameter called the inertia weight, c_1 and c_2 are positive constants referred to as cognitive and social parameters, respectively, $\mathbf{r}_i$, $i = 1, 2$, is a $D \times D$ diagonal matrix whose diagonal elements are random numbers uniformly distributed in the range of $[0, 1]$. A large inertia weight encourages global search while a small value facilitates local exploitation. Therefore, the inertia weight is critical for the search behavior of the PSO, and a good balance between exploration and exploitation can be achieved using a dynamical inertia weight. Experimental results show that it is beneficial to start with a large inertia weight in the early search stage in order to enhance exploration of the search space, and gradually decrease the inertia to achieve more refined solutions in the final search stage. Shi et al. [48] introduced a linearly decreasing inertia weight into the PSO to significantly improve its performance through an empirical study of inertia weight. The inertia weight is adapted over the search iteration t as follows:

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \times t \tag{3}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the initial and final values of the inertia weight, respectively, and $t_{\max}$ is the maximum iteration number. Typically, parameters $\omega_{\max}$ and $\omega_{\min}$ are set to 0.9 and 0.4, respectively. Cognitive and social parameters are acceleration constants where c_1 pushes the particles towards its personal best position and c_2 drives them towards the global best position of the swarm. Both parameters simultaneously control how far a particle will move in a single iteration, and are often set to the same weight [6]. The personal best position of particle i is determined as follows:

$$\vec{p}_i(t) = \begin{cases} \vec{p}_i(t-1) & \text{if } f(\vec{x}_i(t)) > f(\vec{p}_i(t-1)) \\ \vec{x}_i(t) & \text{otherwise} \end{cases} \tag{4}$$

and the global best position can be obtained by:

$$\vec{p}_g(t) \in \{\vec{p}_1(t), \vec{p}_2(t), \ldots, \vec{p}_N(t)\} | f(\vec{p}_g(t)) = \min\{f(\vec{p}_1(t)), f(\vec{p}_2(t)), \ldots, f(\vec{p}_N(t))\} \tag{5}$$

1. **Begin**
2. Parameter setting, including swarm size N, inertia weight ω, cognitive parameter c_1 and social parameter c_2, $t = 0$;
3. Initialize a population with random positions and velocities, evaluate the fitness of each particle;
4. Generate the personal best position of each particle: $\vec{p}_i(t) = \vec{x}_i(t), i = 1, 2, \ldots N$;
5. Generate the global best position of the swarm according to formula (5): find $\min f(\vec{p}_i(t)), i = 1, 2, \ldots, N$, set $\vec{p}_g(t) = \vec{p}_i(t)$ if $f(\vec{p}_i(t))$ is minimal;
6. **While** the stopping criterion is not met
7. **For** each particle i in the swarm
8. Update the velocity and position using (1) and (2);
9. Evaluate the fitness of particle i;
10. Update the personal best position of particle i according to formula (4);
11. **End For**
12. Update the global best position of the swarm according to formula (5): find $\min f(\vec{p}_i(t)), i = 1, 2, \ldots, N$, set $\vec{p}_g(t) = \vec{p}_i(t)$ if $f(\vec{p}_i(t))$ is minimal;
13. t=t+1;
14. **End While**
15. **End**

Fig. 1. Psuedo-code of the standard PSO algorithm.

where f is the objective function and is the swarm size. In PSO, the velocity must be restricted to a range $[\vec{v}_{\min}, \vec{v}_{\max}]$ in order to prevent the particles from flying out of the search space. Commonly, the velocity is set as follows:

$$\vec{v}_i(t+1) = \begin{cases} \vec{v}_{\min} & \text{if } \vec{v}_i(t+1) \leqslant \vec{v}_{\min} \\ \vec{v}_{\max} & \text{if } \vec{v}_i(t+1) \geqslant \vec{v}_{\max} \\ \vec{v}_i(t+1) & \text{otherwise} \end{cases} \tag{6}$$

Fig. 1 shows the pseudo-code of the standard PSO.

Convergence to the global optimum and a fast convergence are two main issues that need to be addressed in order to improve the performance of the PSO. To enhance the capability of PSO to converge to the global optimum, Chen and Li [7] added a random velocity to balance exploration and convergence of PSO. Mei et al. [37] introduced a random solution in PSO as the best solution at the end of every iteration to improve the quality of the obtained solutions. Latiff and Tokhi [27] introduced a new parameter termed as the spread factor to accelerate the convergence speed of PSO. Modiri et al. [38] eliminated the personal best in velocity update to achieve a faster convergence when the search space is large, which was shown to be successful in antenna design.

3. A New Fitness Estimation Strategy for Particle Swarm Optimization (FESPSO)

3.1. The proposed estimation strategy

Theoretically, all particles in PSO fly towards the optimal solution and finally converge to the best position. So it can be imagined that many particles will collide with each other during the search. Suppose particles i and j occupy the same position at iteration $t+1$. Obviously, they have the same fitness value. So if the fitness value of particle $i(f(\vec{x}_i(t+1)))$ is known, $f(\vec{x}_j(t+1)) = f(\vec{x}_i(t+1))$ will be also known, obviating the need for an additional expensive fitness evaluation or estimation of fitness. Furthermore, considering two arbitrary particles i and j in the swarm at iteration $(t+1)$, their new positions can be obtained by the following two formulas, according to Eqs. (1) and (2):

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \omega\vec{v}_i(t) + c_1\mathbf{r}_1(\vec{p}_i(t) - \vec{x}_i(t)) + \vec{c}_2\mathbf{r}_2(\vec{p}_g(t) - \vec{x}_i(t)) \tag{7}$$

$$\vec{x}_j(t+1) = \vec{x}_j(t) + \omega\vec{v}_j(t) + c_1\mathbf{r}'_1(\vec{p}_j(t) - \vec{x}_j(t)) + c_2\mathbf{r}'_2(\vec{p}_g(t) - \vec{x}_j(t)) \tag{8}$$

From Eq. (2), we know that

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{9}$$

$$\vec{x}_j(t) = \vec{x}_j(t-1) + \vec{v}_j(t) \tag{10}$$

thus, we have

$$\vec{v}_i(t) = \vec{x}_i(t) - \vec{x}_i(t-1) \tag{11}$$

$$\vec{v}_j(t) = \vec{x}_j(t) - \vec{x}_j(t-1) \tag{12}$$

Substituting (11), (12) into (7), (8), respectively, we can derive that

$$\vec{x}_i(t+1) = (1+\omega - c_1r_1 - c_2r_2)\vec{x}_i(t) - \omega\vec{x}_i(t-1) + c_1r_1\vec{p}_i(t) + c_2r_2\vec{p}_g(t) \tag{13}$$

$$\vec{x}_j(t+1) = (1+\omega - c_1r'_1 - c_2r'_2)\vec{x}_j(t) - \omega\vec{x}_j(t-1) + c_1r'_1\vec{p}_j(t) + c_2r'_2\vec{p}_g(t) \tag{14}$$

It can be easily found that both new positions are related to its personal best position and the global best position of the swarm $\vec{p}_g(t)$. Thus, according to Eqs. (13) and (14), the relationship between the positions of the two arbitrary particles at iteration $t+1$ can be expressed as:

$$\begin{aligned} &c_2r'_2\vec{x}_i(t+1) - c_2r'_2(1+\omega - c_1r_1 - c_2r_2)\vec{x}_i(t) + c_2r'_2\omega\vec{x}_i(t-1) - c_2r'_2c_1r_1\vec{p}_i(t) \\ &\quad = c_2r_2\vec{x}_j(t+1) - c_2r_2(1+\omega - c_1r'_1 - c_2r'_2)\vec{x}_j(t) + c_2r_2\omega\vec{x}_j(t-1) - c_2r_2c_1r'_1\vec{p}_j(t) \end{aligned} \tag{15}$$

After rearrangement, it becomes

$$\begin{aligned} &c_2r'_2\vec{x}_i(t+1) + c_2r'_2\omega\vec{x}_i(t-1) + c_2r_2(1+\omega - c_1r'_1 - c_2r'_2)\vec{x}_j(t) + c_2r_2c_1r'_1\vec{p}_j(t) \\ &\quad = c_2r_2\vec{x}_j(t+1) + c_2r_2\omega\vec{x}_j(t-1) + c_2r'_2(1+\omega - c_1r_1 - c_2r_2)\vec{x}_i(t) + c_2r'_2c_1r_1\vec{p}_i(t) \end{aligned} \tag{16}$$

In order to understand the fitness estimation strategy better, a temporary variable $\vec{x}_v = (x_{v1}, x_{v2}, \ldots, x_{vD})$, called virtual position, is introduced. The value of $\vec{x}_v$ at iteration $t+1$ is defined as:

$$\begin{aligned} \vec{x}_v(t+1) &= c_2r'_2\vec{x}_i(t+1) + c_2r'_2\omega\vec{x}_i(t-1) + c_2r_2(1+\omega - c_1r'_1 - c_2r'_2)\vec{x}_j(t) + c_2r_2c_1r'_1\vec{p}_j(t) \\ &= c_2r_2\vec{x}_j(t+1) + c_2r_2\omega\vec{x}_j(t-1) + c_2r'_2(1+\omega - c_1r_1 - c_2r_2)\vec{x}_i(t) + c_2r'_2c_1r_1\vec{p}_i(t) \end{aligned} \tag{17}$$

From Eq. (17), the virtual position $\vec{x}_v(t+1)$ at iteration $t+1$ can be considered as an offspring of $\vec{x}_i(t+1), \vec{x}_i(t-1), \vec{x}_j(t)$ and $\vec{p}_j(t)$, so the fitness at the virtual position $f(\vec{x}_v(t+1))$ can be obtained by using a convex combination technique. In other words, $f(\vec{x}_v(t+1))$ can be estimated by a weighted sum of fitness values at these four positions. Similarly, the virtual position $\vec{x}_v(t+1)$ can also be regarded as an offspring of $\vec{x}_j(t+1), \vec{x}_j(t-1), \vec{x}_i(t)$ and $\vec{p}_i(t)$, and fitness $f(\vec{x}_v(t+1))$ can further be estimated by a weighted sum of the fitness values at these four positions.

Suppose $d^i_v(t+1)$, $d^i_v(t-1)$, $d^i_v(t)$, $d^{pj}_v(t+1)$, $d^j_v(t-1)$, $d^j_v(t)$ and $d^{pj}_v(t)$ represent the distance between $\vec{x}_v(t+1)$ and $\vec{x}_i(t+1)$, $\vec{x}_i(t-1)$, $\vec{x}_j(t)$, $\vec{p}_j(t)$, $\vec{x}_j(t+1)$, $\vec{x}_j(t-1)$, $\vec{x}_i(t)$ and $\vec{p}_i(t)$, respectively. Here, all distances are calculated with the Euclidean distance. So if none of distances mentioned above equals to 0, the fitness value of $\vec{x}_v(t+1)$ can be obtained by calculating any one of the following two equations:

$$\vec{f}(\vec{x}_v(t+1)) = \frac{\frac{1}{d^i_v(t+1)} f(\vec{x}_i(t+1)) + \frac{1}{d^i_v(t-1)} f(\vec{x}_i(t-1)) + \frac{1}{d^j_v(t)} f(\vec{x}_j(t)) + \frac{1}{d^{pj}_v(t)} f(\vec{p}_j(t))}{\frac{1}{d^i_v(t+1)} + \frac{1}{d^i_v(t-1)} + \frac{1}{d^j_v(t)} + \frac{1}{d^{pj}_v(t)}} \tag{18}$$

$$f(\vec{x}_v(t+1)) = \frac{\frac{1}{d^j_v(t+1)} f(\vec{x}_j(t+1)) + \frac{1}{d^j_v(t-1)} f(\vec{x}_j(t-1)) + \frac{1}{d^i_v(t)} f(\vec{x}_i(t)) + \frac{1}{d^{pi}_v(t)} f(\vec{p}_i(t))}{\frac{1}{d^j_v(t+1)} + \frac{1}{d^j_v(t-1)} + \frac{1}{d^i_v(t)} + \frac{1}{d^{pi}_v(t)}} \tag{19}$$

As seen in Eqs. (18) and (19), the inverse of the distances is used as the weight. As is known, the smaller the distance, the more important the fitness value of that position will be in calculating fitness value of $\vec{x}_v(t+1)$. Combining Eqs. (18), (19) and eliminating $f(\vec{x}_j(t+1))$, we can get an explicit relationship between $f(\vec{x}_i(t+1))$ and $f(\vec{x}_j(t+1))$.

$$\begin{aligned} f(\vec{x}_j(t+1)) = d^j_v(t+1) \Bigg\{ & \alpha \left[\frac{1}{d^i_v(t+1)} f(\vec{x}_i(t+1)) + \frac{1}{d^i_v(t-1)} f(\vec{x}_i(t-1)) + \frac{1}{d^j_v(t)} f(\vec{x}_j(t)) + \frac{1}{d^{pj}_v(t)} f(\vec{p}_j(t)) \right] \\ & - \frac{1}{d^j_v(t-1)} f(\vec{x}_j(t-1)) + \frac{1}{d^i_v(t)} f(\vec{x}_i(t)) + \frac{1}{d^{pi}_v(t)} f(\vec{p}_i(t)) \Bigg\} \end{aligned} \tag{20}$$

where

$$\alpha = \frac{\frac{1}{d^j_v(t+1)} + \frac{1}{d^j_v(t-1)} + \frac{1}{d^i_v(t)} + \frac{1}{d^{pi}_v(t)}}{\frac{1}{d^i_v(t+1)} + \frac{1}{d^i_v(t-1)} + \frac{1}{d^j_v(t)} + \frac{1}{d^{pj}_v(t)}}$$

Obviously it can be seen that if the fitness value of an arbitrary particle i at iteration $t+1$ is known, fitness value of any other particle j at iteration $t+1$ can be estimated by Eq. (20). It is possible to estimate the fitness value of all other particles using Eq. (20) if the fitness of particle i has been evaluated or estimated. Considering the special case mentioned above, where particle j is at the same position as particle i, the fitness values of these two particles are the same. On the other hand, fitness values of all other particles (which are not at the same position as particle i) can be estimated by using Eq. (20). Evidently, if the fitness of a large number of particles is estimated using Eq. (20), the PSO may be misled. To relieve this problem [20], only the fitness value of the closest neighboring particle j(yet not at the same position as particle i) will be estimated in this paper. The reason is that two closest particles may have similar fitness values, irrespective of whether $f(\vec{x}_i(t+1))$ is estimated or evaluated.

Fig. 2 shows a simple example to illustrate the proposed approach.

In Fig. 2, each black dot represents a particle in the space, and each triangle means more than one particle occupy this position. Suppose that the fitness value of particle i has been obtained by evaluating the objective function or by estimating by Eq. (20) at iteration $t+1$. Since particle is the closest neighbor of particle i, then the fitness value of particle j will be estimated according to Eq. (20), if its fitness has not been evaluated by the expensive objective function. However, it can happen that particle j is the closest particle to more than one particle. As shown in this example, position $\vec{x}_j(t+1)$ has the minimum distance to particles i and k. In this case, assuming we are dealing with minimization problems, the fitness of particle j will be calculated as follows:

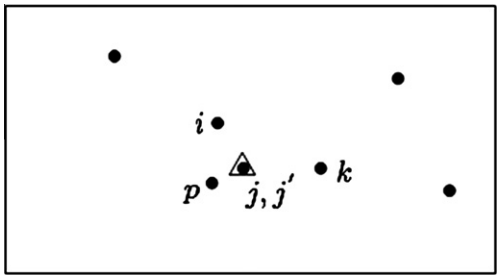

Fig. 2. A simple example.

```
Begin
  Parameter setting, including swarm size N, inertia weight ω, cognitive parameter c1
  and social parameter c2, t = 0;
  Initialize a population with random positions and velocities, evaluate fitness of each
  particle;
  Generate the personal best position of each particle: p_i(t) = x_i(t), i = 1, 2, ..., N;
  fitness(p_i(t)).evaluation = 1; fitness(p_i(t)).estimation = 0; i = 1, 2, ..., N
  Generate the global best position of the swarm: find min f(p_i(t)), i = 1, 2, ..., N, set
  p_g(t) = p_i(t) if f(p_i(t)) is minimal;
  While the stopping criterion is not met
    For each particle i in the swarm
      fitness(x_i(t + 1)).estimation = 0;
      fitness(x_i(t + 1)).evaluation = 0;
    End For
    For each particle i in the swarm
      Update the velocity and position using (1) and (2);
      If (fitness(x_i(t + 1)).estimation = 1 or
        fitness(x_i(t + 1)).evaluation = 1)
        Call the procedure of fitness estimation for the closest particle j
      Else
        Evaluate the fitness of particle i;
        fitness(x_i(t + 1)).estimation = 0;
        fitness(x_i(t + 1)).evaluation = 1;
        Call the procedure of fitness estimation for the closest particle j
      End If
      If f(x_i(t + 1)) ≤ f(p_i(t)) // Update the personal best position of particle i
        p_i(t + 1) = x_i(t + 1)
        fitness(p_i(t + 1)).estimation = fitness(x_i(t + 1)).estimation;
        fitness(p_i(t + 1)).evaluation = fitness(x_i(t + 1)).evaluation;
      Else
        p_i(t + 1) = p_i(t);
        fitness(p_i(t + 1)).estimation = fitness(p_i(t)).estimation;
        fitness(p_i(t + 1)).evaluation = fitness(p_i(t)).evaluation;
      End If
    End For
    Update the global best position of the swarm: Find min f(p_i(t + 1)), i = 1, 2, ...,
    N; suppose f(p_i(t + 1)) is minimal;
    If f(p_i(t + 1)) < f(p_g(t))
      If fitness(p_i(t + 1)).estimation = 1
        Evaluate the fitness of p_i(t + 1);
        fitness(p_i(t + 1)).evaluation = 1;
        f(p_k(t + 1)) = f(p_i(t + 1)), where p_k(t + 1) = p_i(t + 1), k ≠ i
        fitness(p_k(t + 1)).evaluation = fitness(p_i(t + 1)).evaluation;
        If f(p_i(t + 1)) < f(p_g(t))
          p_g(t + 1) = p_i(t + 1);
        Else
          p_g(t + 1) = p_g(t);
        End If
      Else
        p_g(t + 1) = p_i(t + 1);
      End If
    Else
      p_g(t + 1) = p_g(t);
    End If
    t=t+1;
  End While
End
```

Fig. 3. Psuedo-code of the FESPSO algorithm.

$$f(\vec{x}_j(t+1)) = \begin{cases} f_r(\vec{x}_j(t+1)) & \text{if the fitness value of particle } j \\ & \text{has been evaluated by the} \\ & \text{objective function} \\ \min\limits_{i \in A} f_i(\vec{x}_j(t+1)) & \text{otherwise} \end{cases} \quad (21)$$

where i and j represent particles i and j, respectively; $f_r(\vec{x}_j(t+1))$ is the real fitness value of particle j calculated by the expensive objective function; $f_i(\vec{x}_j(t+1))$ is the fitness value of particle j estimated from particle i; is a set where each element represents a particle from which particle j has the minimum distance. Besides, attention should be paid to particles at the same position as particle j. The fitness value of all these particles will be the same as that of particle j. In this example, it is $f_r(\vec{x}_{j'}(t+1)) = f_j(\vec{x}_j(t+1))$.

In addition, it is worth noting that if the virtual position is the same as the position of $\vec{x}_i(t+1), \vec{x}_i(t-1), \vec{x}_j(t), \vec{p}_j(t), \vec{x}_j(t+1), \vec{x}_j(t-1), \vec{x}_i(t)$ or $\vec{p}_i(t)$, it will result in at inability in establishing the relationship between the fitness values of particles $\vec{x}_i(t+1)$ and $\vec{x}_j(t+1)$. Thus, the fitness value of the virtual position will be the fitness value of all particles at that position, obviating the need for additional fitness estimations or evaluations. To simplify fitness estimation, when any of the distance $d_v^i(t+1), d_v^i(t-1), d_v^j(t), d_v^{pj}(t+1), d_v^j(t+1), d_v^j(t-1), d_v^i(t)$ or $d_v^{pj}(t)$ is equal to 0, nothing will be done to the closest particle j.

3.2. Selection of the global best of the swarm

In the standard PSO, the global best position is selected from the best-found particles in the whole swarm. However, in the proposed algorithm, since the fitness of some particles is estimated, the fitness of the global best particle may be estimated and is subject to estimation errors. To ensure a correct convergence to the true optimum, if the fitness of a local best particle $f(p_{best})$ is better than that of the global best $f(g_{best})$, the fitness of the local best particle will be re-evaluated using the

1. **Begin**
2. $f(\vec{x}_k(t+1)) = f(\vec{x}_i(t+1))$ where $\vec{x}_k(t+1) = \vec{x}_i(t+1)$, $k \neq i$;
3. fitness($\vec{x}_k(t+1)$).estimation= fitness($\vec{x}_i(t+1)$).estimation;
4. fitness($\vec{x}_k(t+1)$).evaluation= fitness($\vec{x}_i(t+1)$).evaluation;
5. Find the closest particle j to $\vec{x}_i(t+1)$;
6. **If** (fitness($\vec{x}_j(t+1)$).evaluation=0)
7. Calculate $d_v^i(t+1)$, $d_v^i(t-1)$, $d_v^j(t)$, $d_v^{pj}(t)$, $d_v^j(t+1)$, $d_v^j(t-1)$, $d_v^i(t)$ and $d_v^{pi}(t)$;
8. **If** all these distances are not equal to 0
9. Estimate a fitness for particle j using (20), denoted by $f_i(\vec{x}_j(t+1))$;
10. **If** (fitness($\vec{x}_j(t+1)$).estimation=1)
11. Set $f(\vec{x}_j(t+1)) = \min f(\vec{x}_j(t+1), f_i(\vec{x}_j(t+1)))$;
12. **Else**
13. Set $f(\vec{x}_j(t+1)) = f_i(\vec{x}_j(t+1))$;
14. fitness($\vec{x}_j(t+1)$).estimation=1;
15. **End If**
16. $f(\vec{x}_k(t+1)) = f(\vec{x}_j(t+1))$ where $\vec{x}_k(t+1) = \vec{x}_j(t+1), k \neq j$;
17. fitness($\vec{x}_k(t+1)$).estimation= fitness($\vec{x}_j(t+1)$).estimation;
18. fitness($\vec{x}_k(t+1)$).evaluation= fitness($\vec{x}_j(t+1)$).evaluation;
19. **End If**
20. **End If**
21. **End**

Fig. 4. Psuedo-code of the procedure of fitness estimation.

expensive objective function before the fitness of the global best is replaced. The global best will be updated only if the local best has a better fitness that the global best on the real objective function.

3.3. Description of the algorithm

From Eq. (20), it can be easily seen that the estimated fitness value of particle j at iteration $t+1$ is related to fitness values of particles i and j at iteration t and $t-1$. This means that the fitness of all particles in the first two iterations must be evaluated by the real objective functions.

The pseudo-code of FESPSO algorithm is shown in Figs. 3 and 4 gives the pseudo-code of the procedure of fitness estimation.

In Figs. 3 and 4, $fitness(\vec{x}_i(t+1)).estimation$ is a flag indicating whether the fitness of particle is estimated by Eq. (20). If this value equals to 0, it means that the fitness is evaluated using the real objective function; otherwise estimated from Eq. (20). Similarly, $fitness(\vec{x}_i(t+1)).evaluation$ represents whether the fitness of particle i is evaluated by the real objective function. A value 1 means to evaluate the fitness of this particle using the real objective function.

It should be noted that if more than one particle has the same minimum distance to particle i, one of these particles randomly is chosen to estimate its fitness.

4. Results and discussions

To evaluate the performance of the proposed algorithm, some widely used benchmark problems are adopted in this paper. The benchmarks have been grouped into four classes by Molga and Smutnicki [39]: (a) uni-modal, convex, multi-dimensional; (b) multi-modal, two-dimensional with a small number of local optimums; (c) multi-modal, two-dimensional with a large number of local optimums; and (d) multi-modal, multi-dimensional, with a large number of local optimums. Problems class (a), albeit uni-modal and convex, also contains problems whose global optimum is hard to locate. Class (b) is used to test the performance of optimizers in the hostile environment, where there are a few local optimums with one global optimum. Classes (c) and (d) are more challenging, which are meant to compare more sophisticated optimization algorithms. In this paper, only benchmark class (a), (b), and (d) are taken to evaluate the proposed FESPSO algorithm.

The benchmark functions used in this paper are described as follows:

(1) Sum of Different Power function:

$$f_1(\vec{x}) = \sum_{i=1}^{30} |x_i|^{i+1} \tag{22}$$

where $|x_i| \leqslant 1$, $i = 1, 2, \ldots, 30$.

$$f_1(\vec{x}^*) = f_1(0, 0, \ldots, 0) = 0$$

The Sum of Different Powers function is a commonly used uni-modal test function.

(2) Rosenbrock's Valley problem:

$$f_2(\vec{x}) = 100(x_2^2 - x_1)^2 + (1 - x_1)^2 \tag{23}$$

where $|x_i| \leqslant 100$.

$$f_2(\vec{x}^*) = f_2(1, 1) = 0 \tag{24}$$

The global optimum of the Rosenbrock's function lies inside a long, narrow, parabolic shaped flat valley. It is trivial to find the valley, nevertheless, it is challenging for the optimizer to converge to its global optimum.

(3) Shekel's Foxholes function:

$$f_3(\vec{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}\right]^{-1} \tag{25}$$

where $|x_i| \leqslant 65$, a_{ij} is satisfied with matrix

$$\begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -33 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$$

$$f_3(\vec{x}^*) = f_1(-32, -32) \approx 1.0$$

Shekel's Foxholes function is a multi-modal test function.

(4) Six-Hump Camel-Back function:

$$f_4(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 + 2 \tag{26}$$

where $|x_i| \leqslant 65$.

$f_4(\vec{x}^*) = f_4(0.08983, -0.7126) = f_4(-0.08983, 0.7126) \approx 0.9683715$

The Six-Hump is often used to test the global search ability of an optimization algorithm. Within the given search space, the problem has six local minima, two of which are global.

(5) Goldstein-Price function:

$$\begin{aligned} f_5(\vec{x}) = {} & \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \\ & \times \left[30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right] \end{aligned} \tag{27}$$

where $|x_j| \leqslant 100.0$.

$f_5(\vec{x}^*) = f_5(0, -1) = 3$

The Goldstein-Price function is also a global optimization test function with a small number of local minima.

(6) Generalized Schwefel problem 2.26:

$$f_6(\vec{x}) = 12569.5 - \sum_{j=1}^{30}\left(x_j \sin\left(\sqrt{|x_j|}\right)\right) \tag{28}$$

where $|x_j| \leqslant 500.0$.

$f_6(\vec{x}^*) = f_6(420.9687, 420.9687, \ldots, 420.9687) \approx 0$

The Schwefel's function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minimum. Therefore, a search algorithm is potentially prone to converge in the wrong direction.

(7) Rastrigin's function:

$$f_7(\vec{x}) = \sum_{j=1}^{30}\left(x_j^2 - 10\cos(2\pi x_j) + 10.0\right) \tag{29}$$

where $|x_j| \leqslant 5.12$.

$f_7(\vec{x}^*) = f_7(0, 0, \ldots, 0) = 0$

The Rastrigin's function is based on the function of De Jong with addition of cosine modulation in order to produce multiple local minima. Thus, this test function is highly multi-modal. However, the locations of the minima are regularly distributed.

(8) Generalized Griewank function:

$$f_8(\vec{x}) = \frac{1}{4000}\sum_{j=1}^{30}x_j^2 - \prod_{j=1}^{30}\cos\left(\frac{x_j}{\sqrt{j}}\right) + 1 \tag{30}$$

where $|x_j| \leqslant 600.0$.

$f_8(\vec{x}^*) = f_8(0, 0, \ldots, 0) = 0$

The Griewank function is a function widely used to test the convergence performance of optimization algorithms. It is similar to the Rastrigin's function and has many widespread yet regularly distributed local minima.

The Sum of Different Power function and Rosenbrock's Valley function are continuous, convex and uni-modal with only one local minimum. The Six-Hump Camel-Back and Goldstein-Price are 2-dimensional multi-modal functions with only a few local minima, while Shekel's Foxholes, Generalized Schwefel Problem 2.26, Rastrigin and Griewank are multi-dimensional multi-modal functions with a huge number of local minima.All benchmarks have an explicit objective function and are computationally inexpensive *per se*. Nevertheless we hope that if the proposed algorithm can greatly reduce the number of fitness evaluations to achieve acceptable solutions on these test problem, they are expected to be computationally efficient also on real-world expensive problems of the same type. For expensive problems, the computational time for fitness

evaluations is dominant compared to other computational costs, therefore in this paper, the number of fitness evaluations using the real objective function is used as the baseline for comparison.

The parameters of the PSO in our experiments are set the same as those recommended in [16] that produced the best performance [6]. The inertia weight ω is decreased linearly from 0.9 down to 0.4; the cognitive parameter and social parameter are all set to 2.05; and the size of the swarm is 30. In general, v_{max} is the maximum distance a particle can travel in an iteration, and the performance of the algorithm improves as v_{max} shrinks. However, too small a value of v_{max} degrades the ability of the swarm to explore the search space [6]. Therefore, in our experiments, v_{max} is set to the upper bound for the Sum of Different Power Function, Shekel's Foxholes, Six-Hump Camel-Back and Goldstein-Price functions, which have only one or a few local minima, and upper bound for the Rosenbrock's Valley Problem, which is difficult for an optimizer to converge to its global optimum.For each test problem, 30 independent runs are performed. The maximum number of iteration is set to 1000 for the Sum of Different Power, Rosenbrock's Velley, Shekel's Foxholes, Six-Hump Camel-Back and Goldstein-Price functions, and to 5000 for the Generalized Schwefel Problem 2.26, Rastrigin and Griewank functions.

Table 1 shows the comparative of results obtained by the canonical PSO and the proposed FESPSO with the above experimental settings, where "Opt." represents the optimal solution currently known for each problem, "Best", "Mean" and "Worst" represent the best, the mean and the worst values of optimal solutions achieved in 30 independent runs, respectively. "Std." stands for the standard deviation of the obtained optimal solutions. "Eval. Num." represents the mean number of fitness evaluations of the 30 independent runs by FESPSO algorithm.

From Table 1, it can clearly be seen that there is little difference between the mean optimal results of the 30 independent runs obtained by the canonical PSO and FESPSO. However, the mean numbers of fitness evaluations using the real function needed by FESPSO are much smaller than those required by the canonical PSO. The results obtained by the canonical PSO and FESPSO on functions f_1 and f_2 were very close to the known optimal minima, and the FESPSO algorithm got the same best optimal solutions as canonical PSO, on functions f_3, f_4 and f_5. However, the mean numbers of fitness evaluations required to obtain optimal minima by FESPSO are much smaller than those required by the canonical PSO. Compared with the canonical PSO, FESPSO is able to reduce computational time by 26.44%, 45.99%, 68.15%, 68.08% and 34.18% on test functions f_1, f_2, f_3, f_4 and f_5, respectively. The canonical PSO has been empirically shown to perform well on many optimization problems. However, it may easily get trapped in a local optimum for high dimensional multi-model problems [5]. In our FESPSO algorithm, the fitness estimation strategy is based on the canonical PSO, and no strategies have been taken to prevent the algorithm from falling into local minima, which is the reason why neither the canonical PSO nor FESPSO has found the best known solution for functions f_6, f_7 and f_8 which are multi-dimensional multi-model functions having many local minima. However, the optimal results obtained by the canonical PSO and FESPSO are very close, and FESPSO, compared to the canonical PSO, saves on average 49.62%, 57.62% and 40.34% of needed number of fitness evaluations on test functions f_6, f_7 and f_8, respectively.

Figs. 5–12 show the convergence performance of FESPSO and the canonical PSO, respectively, averaged over 30 runs on functions $f_1, f_2, f_3, f_4, f_5, f_6, f_7$ and f_8. It can be seen from Figs. 5–12 that FESPSO can obtain comparable optimal solutions on all these functions except that it obtained much better solutions on the generalized Schwefel problem 2.26 (f_6). In addition, the convergence speed of FESPSO is almost the same as or faster than that of the canonical PSO on all test functions studied in this paper, except the Sum of Different Power Function (f_1). It is not difficult to conceive that for continuous, convex and unimodal functions having only one local minimum (f_1 and f_2), the convergence speed of FESPSO is not superior to that of the canonical PSO, while on all multi-modal functions, FESPSO converges much faster. Our findings confirm the results reported

Table 1
Experimental results on eight functions.

		Opt.	Best	Mean	Worst	Std.	Eval. num.
f_1	PSO	0.000000e+000	5.708428e−009	8.786653e−009	9.925682e−009	2.181225e−009	21453
	FESPSO	0.000000e+000	4.787604e−009	8.509718e−009	9.917596e−009	2.422543e−009	15780
f_2	PSO	0.000000e+000	9.293577e−011	6.207485e−009	9.976654e−009	5.741965e−010	18396
	FESPSO	0.000000e+000	2.891232e−011	6.218043e−009	9.993585e−009	5.246526e−010	9935
f_3	PSO	1.000000e+000	1.000000e+000	1.000000e+000	1.000000e+000	2.026981e−017	30000
	FESPSO	1.000000e+000	1.000000e+000	1.000000e+000	1.000000e+000	1.782815e−013	9554
f_4	PSO	9.683715e−001	9.683715e−001	9.683715e−001	9.683715e−001	6.080942e−017	30000
	FESPSO	9.683715e−001	9.683715e−001	9.683715e−001	9.683715e−001	6.420158e−012	9577
f_5	PSO	3.000000e+000	3.000000e+000	3.000000e+000	3.000000e+000	6.241275e−010	12990
	FESPSO	3.000000e+000	3.000000e+000	3.000000e+000	3.000000e+000	5.297530e−010	8550
f_6	PSO	0.000000e+000	4.224507e+003	6.103523e+003	7.501716e+003	1.625244e+002	150000
	FESPSO	0.000000e+000	4.125744e+003	5.587733e+003	6.948875e+003	1.298683e+002	75567
f_7	PSO	0.000000e+000	2.089417e+001	3.578538e+001	5.870261e+001	1.703083e+000	150000
	FESPSO	0.000000e+000	1.890609e+001	3.866891e+001	6.467473e+001	2.302950e+000	64105
f_8	PSO	0.000000e+000	8.281503e−009	1.303764e−002	1.713657e−002	2.201025e−003	140638
	FESPSO	0.000000e+000	4.417268e−006	2.452592e−002	7.844721e−002	4.252773e−003	80124

$f_1, f_2, f_3, f_4, f_5, f_6, f_7$ and f_8 respectively represent Different Power function, Rosenbrock's Valley problem, Shekel's Foxholes function, Six-hump Camel-Back function, Goldstein-Price function, Generalized Schwefel problem 2.26, Rastrigin's function and Generalized Griewank function.

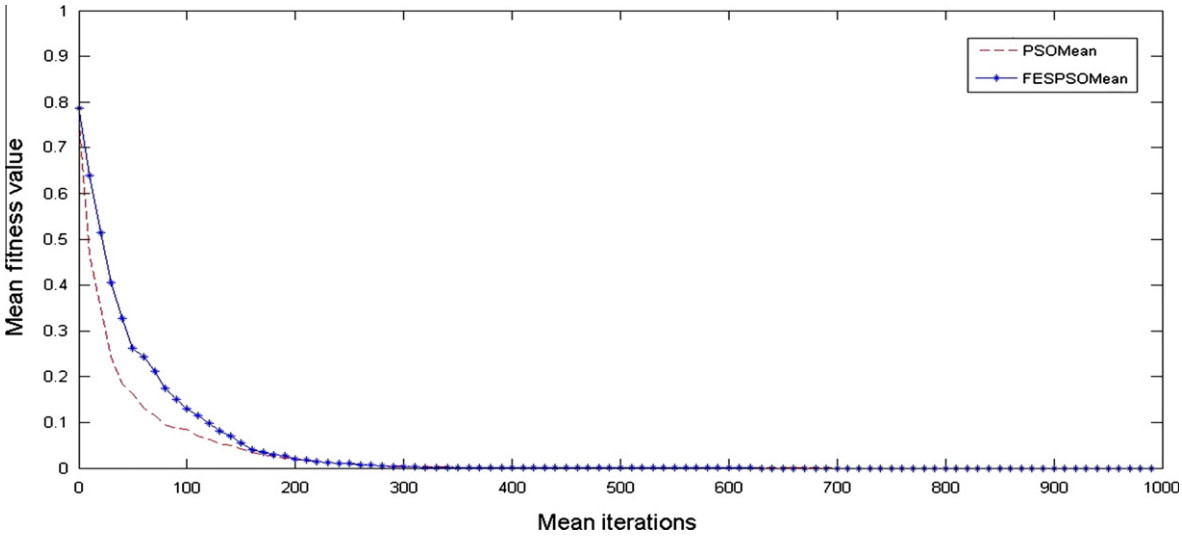

Fig. 5. Performance comparison for Sum of Different Power function (f_1).

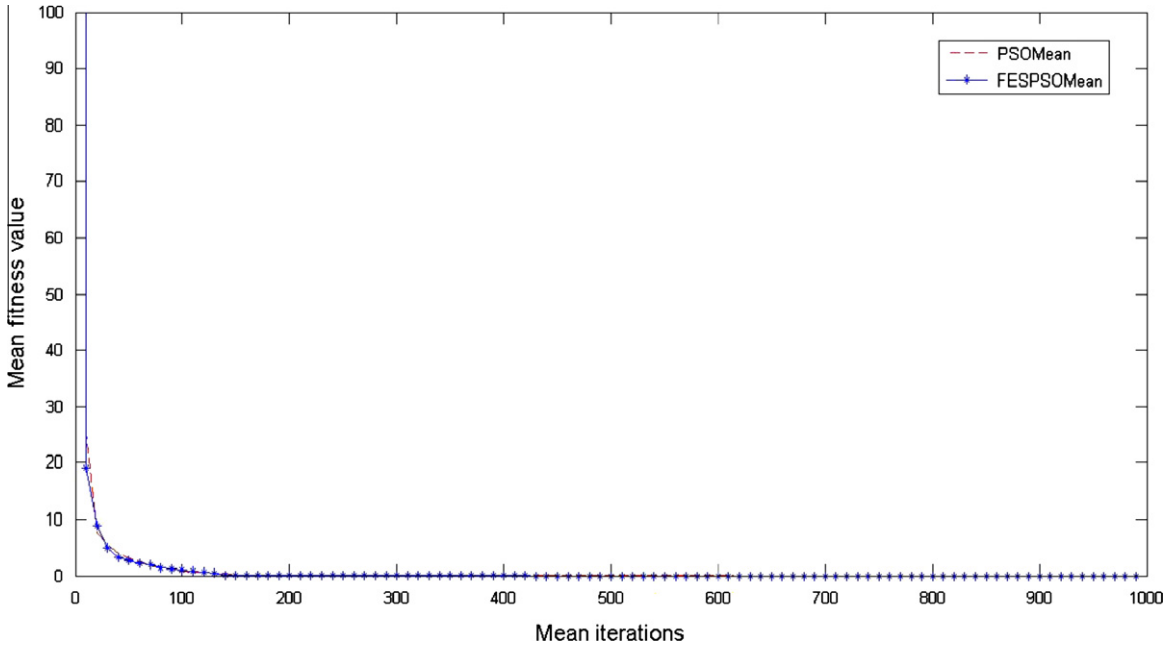

Fig. 6. Performance comparison for Rosenbrock's Valley problem (f_2).

in Lim et al. [33] that approximation errors introduced in fitness estimations may smooth the rugged fitness landscape of multi-modal problems, thereby speeding up the convergence.

To further examine the performance of FESPSO, FPSO3 and FPSO4 proposed in [11] are also compared to the proposed approach. Cui et al. [11] proposed four PSO variants. The first, termed FPSO1, combines the canonical PSO with a traditional convex combination technique. Random fitness evaluations are introduced into resulting in a second variant, called FPSO2. FPSO2 is further combined with additional convex combination techniques using two different reliability updating methods on the global best position, where, in the third variant, FPSO3, a reliability value is linearly increased from 0.1 to 1, whereas in the fourth variant, FPSO4, the reliability value is randomly chosen from $(0,1)$. The empirical results obtained by Cui et al. [11] indicated that the performance of FPSO1 and FPSO2 is worse than or comparable to that of FPSO3 and FPSO4. So in this paper, only FPSO3 and FPSO4 are compared with FESPSO. Parameters settings for FPSO3 and FPSO4 are the same as for FESPSO, except that v_{max} is set to 10% of the upper bound of domain and the swarm size is set to 100.

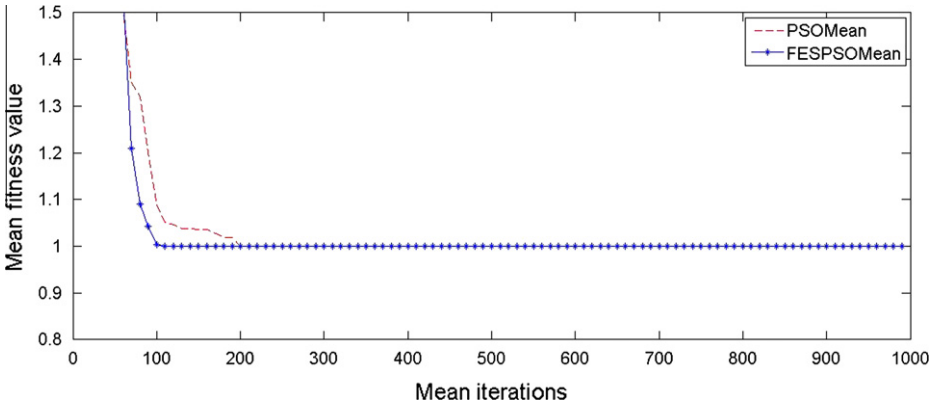

Fig. 7. Performance comparison for Shekel's Foxholes function (f_3).

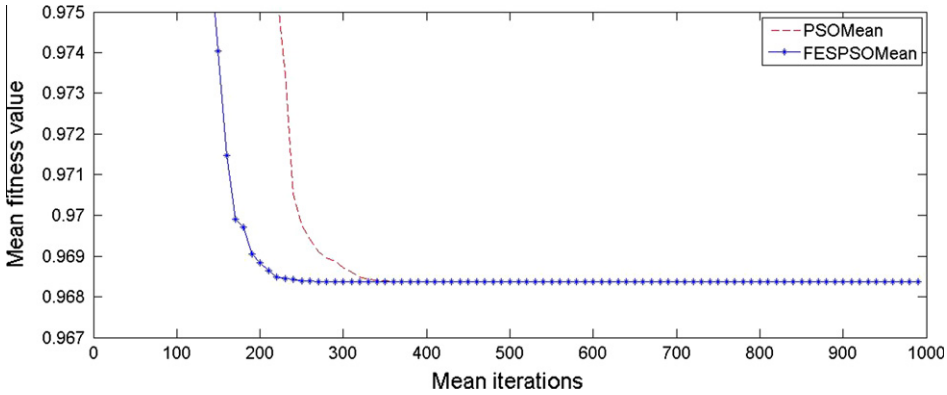

Fig. 8. Performance comparison for Six-Hump Camel-Back function (f_4).

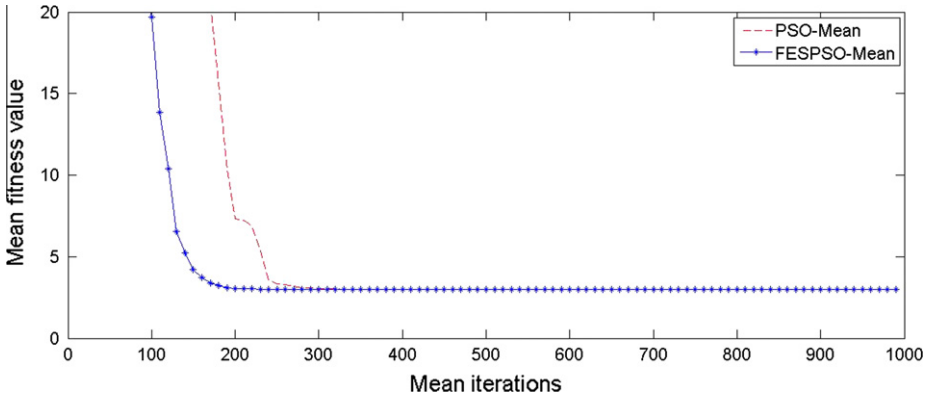

Fig. 9. Performance comparison for Goldstein-Price function (f_5).

Tables 2–7 show the comparative results on six test functions f_3, f_4, f_5, f_6, f_7 and f_8. f_1 and f_2 are not included because the optimal results for these two functions were not provided in [11]. It can be seen from Tables 2–4 that on multi-model test functions having only a few local minima, FESPSO not only obtained the best optimal solutions that are no worse than those obtained by FPSO3 and FPSO4, but also needed a much smaller number of fitness evaluations on average than FPSO3 and FPSO4.

For functions having many local minima, as seen from Table 5, FESPSO obtained better mean fitness values with fewer fitness evaluations on average than FPSO3 and FPSO4 on the Generalized Schwefel Problem 2.26. Comparisons of FESPSO with FPSO3 on Rastrigin's Function and Generalized Griewank Function (see Tables 6 and 7) show that although the mean fitness values obtained by FPSO3 are 31.45% and 16%, respectively, better than those of FESPSO, the average numbers of fit-

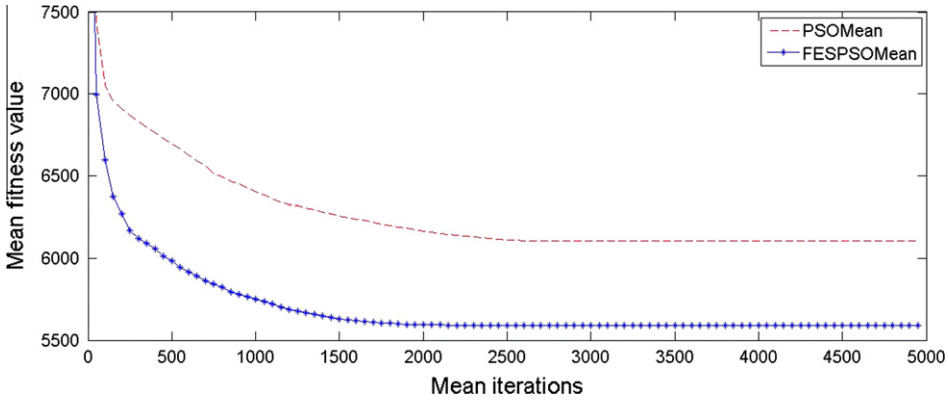

Fig. 10. Performance comparison for Generalized Schwefel problem 2.26 (f_6).

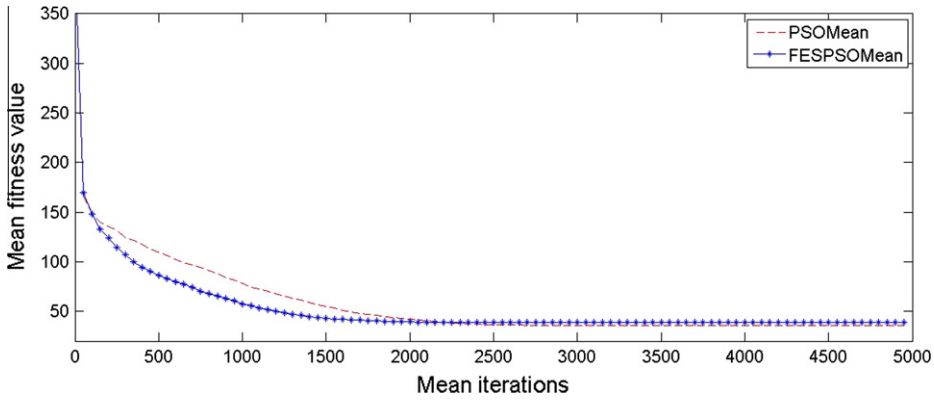

Fig. 11. Performance comparison for Rastrigin's function (f_7).

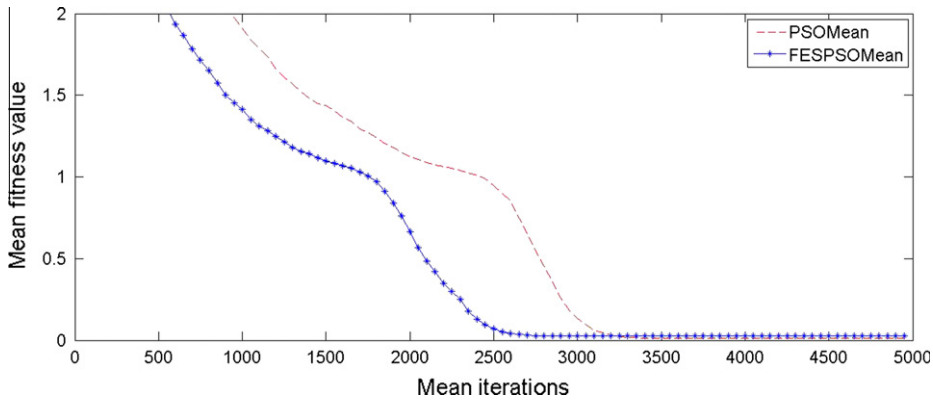

Fig. 12. Performance comparison for Generalized Griewank function (f_8).

ness evaluations used by FESPSO are 38.97% and 23.46% smaller than those of FPSO3. Comparison of FESPSO with FPSO4 on Rastrigin's Function (Table 6) shows that the mean fitness value of FESPSO is 36.85% worse than FPSO4 and the mean number of fitness evaluations of FESPSO is 34.66% less than that of FPSO4. However, for Generalized Griewank Function, although the mean fitness value obtained by FPSO4 is 2.74% better than that obtained by FESPSO, the mean number of fitness evaluations FESPSO needed is 17.24% less than FPSO4.

On the whole, the performance of FESPSO on reduction of fitness evaluations is better than that of FPSO3 and FPSO4. Its capability of obtaining the optimal solution is also better than that of FPSO3 and FPSO4 on the multi-modal test functions

Table 2
Comparison results for Shekel's Foxholes function.

Algorithm	Mean	Std.	Eval. num.
FPSO3 [11]	1.228634e+000	9.227708e−001	17702
FPSO4 [11]	1.296080e+000	5.313548e−001	16253
FESPSO	1.000000e+000	1.782815e−013	9554

Table 3
Comparison results for Six-Hump Camel-Back function.

Algorithm	Mean	Std.	Eval. num.
FPSO3 [11]	9.683715e−001	1.376362e−015	17997
FPSO4 [11]	9.683715e−001	1.714257e−015	15991
FESPSO	9.683715e−001	6.420158e−012	9577

Table 4
Comparison results for Goldstein-Price function.

Algorithm	Mean	Std.	Eval. num.
FPSO3 [11]	3.000000e+000	6.900034e−015	17190
FPSO4 [11]	3.000000e+000	5.159850e−015	15306
FESPSO	3.000000e+000	5.297530e−010	8550

Table 5
Comparison results for Generalized Schwefel problem 2.26.

Algorithm	Mean	Std.	Eval. num.
FPSO3 [11]	5.859118e+003	6.752567e+002	106659
FPSO4 [11]	5.809730e+003	6.156246e+002	99042
FESPSO	5.587733e+003	1.298683e+002	75567

Table 6
Comparison results for Rastrigin's function.

Algorithm	Mean	Std.	Eval. num.
FPSO3 [11]	2.941767e+001	7.530736e+000	105043
FPSO4 [11]	2.825683e+001	8.314577e+000	98105
FESPSO	3.866891e+001	2.302950e+000	64105

Table 7
Comparison results for Generalized Griewank function.

Algorithm	Mean	Std.	Eval. num.
FPSO3 [11]	2.114311e−002	1.710436e−002	104676
FPSO4 [11]	2.387290e−002	2.191084e−002	96813
FESPSO	2.452592e−002	4.252773e−003	80124

having a small number of local optimums, it performed worse on the multi-modal test functions having a large number of local optimums investigated in this paper.

5. Conclusions

A new fitness estimation strategy, FESPSO is proposed to reduce the computational cost for optimization of computationally-expensive problems using particle swarm optimization. In the proposed algorithm, fitness estimation is based on an explicit formula derived from the position and velocity updating rules in PSO, which is computationally very efficient. Unlike most surrogate-assisted evolutionary algorithms where the same surrogate is used for all individuals, the approximation models in FESPSO are different for different particles, and the updating of the approximation model is done in every iteration,

which makes the fitness estimation more reliable. As a whole, FESPSO outperforms the compared PSO variants on most test problems used in this study, although it performed less well on multi-modal functions having a large number of local minima compared to some sophisticated PSO variants. Recall however, that the main purpose of FESPSO is to reduce the number of fitness evaluations in comparison with the canonical particle swarm optimization without seriously deteriorating its search performance, which is of particular important for PSO to be applied to real-world expensive problems. From these perspectives, the proposed FESPSO has successfully achieved its goal demonstrated by experimental results on the test problems examined in this work.

Despite the success the proposed FESPSO has achieved, future work is still required to further reduce computational cost and to improve its search performance on high-dimensional multi-modal test problems. The following two main endeavors can be promising in achieving the above aims:

(1) A reliability measure that measures the confidence of the estimated fitness will be introduced in the proposed FESPSO approach. The reliability in fitness estimation can depend on two factors: the distance between two particles and the reliability of the particle whose fitness value is known. If the particle's fitness reliability drops below a threshold, the estimated fitness will be discarded and its fitness will be evaluated using the real objective function, whose reliability becomes the maximum.
(2) The fitness of more particles can be estimated, e.g., with the help of a clustering technique to further reduce the fitness calculation time. The basic idea is to categorize the swarm into a number of sub-swarm, similar to [22], and only the fitness of the particle closest to the center of the sub-swarm will be evaluated using the real objective function, and the fitness of all other particles in the same sub-swarm can be calculated by the proposed estimation formula.

Acknowledgements

This work was supported in part by Youth Foundation of Shanxi Province of China under Grant No. 2011021019-3, by Youth Foundation of Taiyuan University of Science and Technology under Grant No. 20103012 and by Tian Yuan Special Funds of the National Natural Science Foundation of China under Grant No. 11126076.

References

[1] D. Büche, N.N. Schraudolph, P. Koumoutsakos, Accelerating evolutionary algorithms with Gaussian process fitness function models, IEEE Transactions on System, Man, and Cybernetics – Part C: Applications and Reviews 35 (2) (2005) 183–194.
[2] J. Barrera, C.A.C. Coello, A particle swarm optimization method for multimodal optimization based on electrostatic interaction, in: Proceedings of the 8th Mexican International Conference on Artificial Intelligence, Guanajuato, México, 2009, pp. 622–632.
[3] Z. Bouzarkouna, A. Auger, D.Y. Ding, Investigating the local-meta-model CAM-ES for large population sizes, Lecture Notes in Computer Science 6024 (2010) 402–411.
[4] J. Cai, W.D. Pan, On fast and accurate block-based motion estimation algorithms using particle swarm optimization, Information Sciences 197 (15) (2012) 53–64.
[5] X. Cai, et al., Chaper 5: Individual parameter selection strategy for particle swarm optimization, Particle swarm optimization, I-Tech Education and Publishing, Vienna, Austria, 2009, pp. 89–112.
[6] A. Carlisle, G. Dozier, An off-the-shelf PSO, Particle swarm Optimization Workshop, 2001, pp. 1–6.
[7] X. Chen, Y. Li, A modified PSO structure resulting in high exploration ability with convergence guaranteed, IEEE Transactions on system, Man, and Cybernetics – Part B: Cybernetics 37 (5) (2007) 1271–1289.
[8] C.T. Cheng, C.P. Ou, K.W. Chau, Combining a fuzzy optimal model with a genetic algorithm to solve multiobjective rainfall–runoff model calibration, Journal of Hydrology 268 (1–4) (2002) 72–86.
[9] J. Chuanwen, E. Bompardb, A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation, Mathematics and Computers in Simulation 68 (1) (2005) 57–65.
[10] F. Cui, J. Zeng, An effective intelligent algorithm for stochastic optimization problem, in: Proceedings of Chinese Control and Decision Conference, 2009, pp. 3197–3202.
[11] Z. Cui, J. Zeng, G. Sun, A fast particles swarm optimization, International Journal of Innovative Computing, Information and Control 2 (6) (2006) 1365–1380.
[12] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Information Sciences 178 (15) (2008) 3096–3109.
[13] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[14] R. Eberhart, Y. Shi, A modified particle swarm optimizer, in: Proceedings of International Conference on Evolutionary Computation, IEEE Service Center, Anchorage, AK, Piscataway, NJ, 1998, pp. 69–73.
[15] R.C. Eberhart, Y. Shi, Evolving artificial neural networks, in: Proceedings of the 1998 International Conference on Neural Networks and Brain, Beijing, China, 1998, pp. 5–13.
[16] R.C. Eberhat, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 Congress on Evolutionary Computing, 2000, pp. 84–88.
[17] M. Farina, A neural network based generalized response surface multiobjective evolutionary algorithms, in: Proceedings of Congress on Evolutionary Computation, 2002, pp. 956–961.
[18] J.C. Fu, L. Wang, A random-discretization based monte carlo sampling method and its applications, Methodology and Computing in Applied Probability 4 (5–25) (2002).
[19] P. Hajela, J. Lee, Genetic algorithm in multidisciplinary rotor blade design, in: Proceedings of the 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Material Conference, New Orleans, LA, 1995, pp. 2187–2197.
[20] T. Hendtlass, Fitness estimation and the particle swarm optimisation algorithm, in: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 4266–4272.
[21] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing 9 (1) (2005) 3–12.

[22] Y. Jin, B. Sendhoff, Reducing fitness evaluations using clustering techniques and neural network ensembles, in: Proceedings of Genetic and Evolutionary Computation Conference, Seattle, 2004, pp. 688–699.
[23] V.R. Joseph, Y. Hung, A. Sudjianto, Blind kriging: A new method for developing metamodels, Journal of Mechanical Design 130 (3) (2008) 031102.1–031102.8.
[24] M. Kazemi, et al., Constraint importance mode pursuing sampling for continuous global optimization, in: Proceedings of the ASME 2010 International Design Engineering Technical Conference & Computers and Information in Engineering Conference, Montreal, Quebec, Canada, 2010, pp. 1–10.
[25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International. Conference on Neural Networks, 1995, pp. 1942–1948.
[26] J. Kennedy, R. Eberhart, Y. Shi, Swarm intelligence, Morgan Kaufmann division of Academic Press, 2001.
[27] I.A. Latiff, M.O. Tokhi, Fast convergence strategy for particle swarm optimization using spread factor, in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 2693–2700.
[28] M. Li, An improved kriging-assisted multi-objective genetic algorithm, Journal of Mechanical Design 133 (2011) 1–12.
[29] Q. Li et al, Swarm intelligence clustering algorithm based on attractor, Lecture Notes in Computer Science 3612 (2005) 496–504.
[30] Y. Li et al, Dynamic optimal reactive power dispatch based on parallel particle swarm optimization algorithm, Computers & Mathematics with Applications 57 (11–12) (2009) 1835–1842.
[31] Y. Li et al, Optimal reactive power dispatch using particle swarms optimization algorithm based Pareto optimal set, Lecture Notes in Computer Science 5553 (2009) 152–161.
[32] Y. Lian, M.-S. Liou, Multiobjective optimization using coupled response surface model and evolutionary algorithm, AIAA Journal 43 (6) (2005) 1316–1325.
[33] D. Lim et al, Generalizing surrogate-assisted evolutionary computation, IEEE Transactions on Evolutionary Computation 14 (3) (2010) 329–355.
[34] J.Y. Lin, K.W. Chau, Using support vector machines for long-term discharge prediction, Hydrological Sciences Journal 51 (4) (2006) 599–612.
[35] H. Lu, W. Chen, Self-adaptive velocity particle swarm optimization for solving constrained optimization problems, Journal of Global Optimization 41 (3) (2008) 427–445.
[36] M. Ma, L.B. Zhang, Particle swarm optimization algorithm design for fuzzy neural network, Advances in Soft Computing 40 (2007) 309–314.
[37] C. Mei, G. Liu, X. Xiao, Improved particle swarm optimization algorithm and its global convergence analysis, in: Proceedings of the 2010 Chinese Control and Decision Conference, 2010.
[38] A. Modiri, K. Kiasaleh, Modification of real-number and binary pso algorithms for accelerated convergence, IEEE Transactions on Antennas and Propagation 59 (1) (2011) 214–224.
[39] M. Molga, C. Smutnicki, Test functions for optimization needs, Computer and Information Science (2005) 1–43.
[40] N. Muttil, K.W. Chau, Neural network and genetic programming for modelling coastal algal blooms, International Journal of Environment and Pollution 28 (3–4) (2006) 223–238.
[41] Y.-S. Ong, P.B. Nair, K.Y. Lum, Max-min surrogate-assisted evolutionary algorithm for robust design, IEEE Transactions on Evolutionary Computation 10 (4) (2006) 392–404.
[42] Y.S. Ong, A.J. Keane, Meta-lamarckian learning in memetic algorithms, IEEE Transactions on Evolutionary Computation 8 (2) (2004) 99–110.
[43] M. Reyes-Sierra, C.A.C. Coello, A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, 2005, pp. 65–72.
[44] M. Salami, T. Hendtlass, A fast evaluation strategy for evolutionary algorithms, Applied Soft Computing 2/3F (2003) 156–173.
[45] A. Shahrokhi, A. Jahangirian, A surrogate assisted evolutionary optimization method with application to the transonic airfoil design, Engineering Optimization 42 (6) (2010) 497–515.
[46] S. Shan, G.G. Wang, Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions, Structural and Multidisciplinary Optimization 41 (2010) 219–241.
[47] H. Shen et al, A mountain clustering based on improved pso algorithm, Lecture Notes in Computer Science 3612 (2005) 477–481.
[48] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Proceedings of the 7th International Conference on Evolutionary Programming VII, 1998, pp. 591–600.
[49] M.R. Sierra, C.A.C. Coello, Improving PSO-based multi-objective optimization using crowding, mutation and $\in$-dominance, in: Proceedings of the Third International Conference on Evolutionary Multi-criterion Optimization, Guanajuato, Mexico, 2005, pp. 505–519.
[50] C. Sun, J. Zeng, J. Pan, An improved particle swarm optimization with feasibility-based rules for constrained optimization problems, in: Proceedings of the 22nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2009, pp. 202–211.
[51] C. Sun, J. Zeng, J. Pan, A new vector particle swarm optimization for constrained optimization problems, in: Proceedings of the International Joint Conference on Computational Sciences and Optimization, 2009, pp. 485–488.
[52] C. Sun, J. Zeng, J. Pan, An improved vector particle swarm optimization for constrained optimization problems, Information Sciences 181 (6) (2011) 1153–1163.
[53] H. Wang et al, Integrating least square support vector regression and mode pursuing sampling optimization for crashworthiness design, Journal of Mechanical Design 133 (2011).
[54] L. Wang, S. Shan, G.G. Wang, Mode-pursuing sampling method for global optimization on expensive black-box functions, Engineering Optimization 36 (4) (2004) 419–438.
[55] C.L. Wu, K.W. Chau, Y.S. Li, Predicting monthly streamflow using data-driven models coupled with data-preprocessing techniques, Water Resource Research 45 (W08432) (2009).
[56] J.X. Xie, C.T. Cheng, K.W. Chau, A hybrid adaptive time-delay neural network model for multi-step-ahead prediction of sunspot activity, International Journal of Environment and Pollution 28 (3–4) (2006) 364–381.
[57] E. Zahara, Y.T. Kao, Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems, Expert Systems with Applications 36 (2) (2009) 3880–3886.
[58] A.E.M. n. Zavala, A.H. a. Aguirre, E.R.V. Diharce, Particle evolutionary swarm optimization algorithm (peso), in: Proceedings of the Sixth Mexican International Conference on Computer Science, 2005, pp. 282–289.
[59] J. Zhang, K.W. Chau, Multilayer ensemble pruning via novel multi-sub-swarm particle swarm optimization, Journal of Universal Computer Science 15 (4) (2009) 840–858.
[60] Y. Zhang, D. Gong, Z. Ding, A bare-bones multi-objective particle swarm optimization algorithm for environmental/economic dispatch, Information Sciences 192 (1) (2012) 213–227.