Applied Mathematics and Computation 219 (2013) 5185–5197

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

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

Particle swarm optimization using dimension selection methods

Xin Jin [a,b,c,*], Yongquan Liang [b], Dongping Tian [a,c], Fuzhen Zhuang [a]

[a] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[b] College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China
[c] Graduate University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Keywords:
Particle swarm optimization (PSO)
Deterministic particle swarm optimization
Randomness
Random dimension selection
Deterministic dimension selection

ABSTRACT

Particle swarm optimization (PSO) has undergone many changes since its introduction in 1995. Being a stochastic algorithm, PSO and its randomness present formidable challenge for the theoretical analysis of it, and few of the existing PSO improvements have make an effort to eliminate the random coefficients in the PSO updating formula. This paper analyzes the importance of the randomness in the PSO, and then gives a PSO variant without randomness to show that traditional PSO cannot work without randomness. Based on our analysis of the randomness, another way of using randomness is proposed in PSO with random dimension selection (PSORDS) algorithm, which utilizes random dimension selection instead of stochastic coefficients. Finally, deterministic methods to do the dimension selection are proposed, and the resultant PSO with distance based dimension selection (PSODDS) algorithm is greatly superior to the traditional PSO and PSO with heuristic dimension selection (PSOHDS) algorithm is comparable to traditional PSO algorithm. In addition, using our dimension selection method to a newly proposed modified particle swarm optimization (MPSO) algorithm also gets improved results. The experiment results demonstrate that our analysis about the randomness is correct and the usage of deterministic dimension selection method is very helpful.

1. Introduction

Particle swarm optimization (PSO) is a relatively new kind of global search method. This derivative-free method is particularly suited to continuous variable problems and has received increasing attention in the optimization community. PSO is originally developed by Kennedy and Eberhart [1] and inspired by the paradigm of birds flocking. PSO consists of a swarm of particles and each particle represents a potential solution for a problem. Every particle flies through the multi-dimensional search space with a velocity, which is constantly updated by the particle's previous best position and by the global best position found by the whole swarm so far. PSO has been used in many applications as it can be easily implemented and is computationally inexpensive. Many researchers have done a lot of work on the improvements, theoretical analysis and applications of PSO [2,3]. As known to all, when a particle learns from its personal best positions and the global best solution, there is randomness in the form of random coefficients. Little work has been done to analyze the importance of the randomness and the randomness places great challenges for the theoretical analysis of PSO. This paper tries to look into the randomness of PSO and to eliminate the randomness, which is favorable for the theoretical analysis of PSO and for a better understanding of PSO, even for the improvements of PSO.

[*] Corresponding author at: Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China.
E-mail addresses: sdjinxin@gmail.com, jinx@ics.ict.ac.cn (X. Jin).

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.11.020

Simple as it seems, the PSO presents formidable challenge for the people who want to understand it through theoretical analysis. A fully comprehensive mathematical model of particle swarm optimization is still not available by now. One reason for this is that the forces are stochastic, which means the use of standard mathematical tools used in the analysis of dynamical systems is impossible. Another reason is that the PSO is made up of a large number of interacting particles. Although each particle's behavior is simple, understanding the dynamics of the whole is nontrivial. Because of these difficulties, most researchers have often been forced to make simplified assumptions in order to obtain models that could be studied mathematically. Ozcan and Mohan [4] presented a simplified model where the behavior of one particle, in isolation, in one dimension, in the absence of randomness, and during stagnation was considered. Similar assumptions were used by Clerc and Kennedy's model [5], which includes one particle, one dimension and deterministic behavior. Yasuda et al. [6] studied a model with a one-dimensional particle, stagnation, and absence of randomness. Inertia was included in this model. Brandstatter and Baumgartner [7] gave a model using the notions of damping factor and natural vibratory frequency, however, randomness is also not considered in this model. An uniform model of PSO was described and the convergence was analyzed with linear control theory in [8]. Hu and Li [9] proposed and analyzed a simple PSO without particle velocity. Most of the theoretical models did not consider the randomness of PSO, so the conclusions drawn from these models tend to be approximate.

Many adjustments and improvements have been made to the basic PSO algorithm over the past decade to deal with the premature convergence problem and to get a high search speed. As the parameters used in PSO are believed to have great influence on the performance of the algorithm, many researches [10–12] had been done to adaptively adjust the parameters for different problems. PSO is also hybridized with other methods [12–17] to enhance its ability on a larger number of applications. Unlike the conventional PSO which have identical particles, some researches [18,19] proposed heterogeneous PSO in which the particles have different behaviors. The information dissemination of several neighborhood topologies is analyzed theoretically and a PSO with varying topology is proposed in [20]. To avoid the premature convergence problem, the methods to keep the swarm diversity were proposed [21,22]. Zhan et al. [23] gives a thorough analysis of the evolution process of PSO and proposed a novel adaptive algorithm. Several approaches to improve the performance of the PSO were described in [24]. Some of the previous methods improved the general performance of PSO and others improved performance on particular kinds of problems, but they all did not analyze the effects of the randomness, let alone the efforts to eliminate the randomness.

This paper first analyzes the importance and effects of the randomness in the canonical PSO and points out that the randomness cannot be eliminated by simply replace the random coefficients with their expectations, which is used in many simplified theoretical models of PSO. Then, a new method to randomly select some dimensions in the search space for each particle to evolve without the stochastic coefficient is proposed, and this method has comparable performance on ten different benchmark functions compared with the canonical PSO. To further reduce the randomness in the selection of the dimensions that need to be updated, deterministic methods to do the dimension selection based on the swarm's characteristics are studied, and it is expected that these methods will reduce the complexity of the PSO algorithm, even improve the performance of the algorithms as well.

The rest of this paper is organized as follows: Section 2 describes the traditional PSO algorithm. In Section 3, the importance of the randomness in the PSO and its reasons are given. Then four PSO variants are given to demonstrate the correctness of our analysis and to show the methods to the dimension selection. Experiments and results analysis are presented in Section 4. Finally, Section 5 concludes this paper.

2. Particle swarm optimization algorithm

A standard particle swarm optimization maintains a swarm of particles which are flying with some velocities in the n-dimensional search space. The particles have no weight and no volume. The velocity of each particle is determined by the particle's cognitive knowledge, that is its personal best position, and social knowledge, i.e., the best position found by the whole swarm so far. The current position of each particle is represented by $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D is the dimension of the search space. The personal best position of each particle is $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$, and the current velocity of each particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The global best position found by the whole population so far is $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$. To construct the search course, for each particle we update its velocity V_i and position X_i through each variable dimension d using Eqs. (1) and (2) as follows:

$$v_{id}^{t+1} = v_{id}^{t} + c_1 \times r_{1d}^{t} \times \left(pbest_{id}^{t} - x_{id}^{t}\right) + c_2 \times r_{2d}^{t} \times \left(gbest_{d}^{t} - x_{id}^{t}\right), \tag{1}$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1}. \tag{2}$$

In Eq. (1), c_1 is a coefficient for individual confidence, c_2 is a coefficient for social confidence, and r_{1d}^{t} and r_{2d}^{t} are random real numbers which are uniformly distributed on $[0,1]$. Clerc and Kennedy [5] proposed a modified form of PSO that has constriction factor. The velocity update formula is illustrated in Eqs. (3) and (4).

$$v_{id}^{t+1} = \chi \times \left(v_{id}^{t} + c_1 \times r_{1d}^{t} \times \left(pbest_{id}^{t} - x_{id}^{t}\right) + c_2 \times r_{2d}^{t} \times \left(gbest_{d}^{t} - x_{id}^{t}\right)\right), \tag{3}$$

$$\chi = 2 \Big/ \left(\left| 2 - (c_1 + c_2) - \sqrt{(c_1 + c_2)^2 - 4 \times (c_1 + c_2)} \right| \right). \tag{4}$$

According to Clerc's constriction method, c_1 and c_2 are set to 2.05, respectively, and the constriction factor, χ, is approximately 0.7298. The initial swarm and their velocities are often randomly generated, the initialization methods used in this paper are as follows. If the search space is $[l, h]^D$, then every particle's every dimension's position is randomly selected according to a uniform distribution on $[l, h]$. Velocities are generated similarly, if the maximum velocity limit is *Vmax*, then every particle's every dimension's velocity is randomly selected according to a uniform distribution on $[-Vmax, Vmax]$. We randomly generate 1000 particles using the above method, and then choose a number of particles with best fitness to form the initial swarm, the number of particles is often given by the user. The specific algorithm of the PSO with constriction factor is shown in Algorithm 2.

Algorithm 1. Update fitness for the swarm

```
Input:
  $S$: a swarm $S$ of particles,
  $f(x)$: fitness function.
Method:
1: for all particle $x_i$ in swarm $S$ do
2:   Evaluate the particle's fitness using function $f(x_i)$;
3:   Denote the particle's personal best position as $pbest_i$, the swarm's global best position as $gbest$; {suppose it is a
     minimization problem};
4:   if $f(x_i) < f(pbest_i)$ then
5:     $pbest_i \leftarrow x_i$
6:     if $f(x_i) < f(gbest)$ then
7:       $gbest \leftarrow x_i$
8:     end if
9:   end if
10: end for
```

Algorithm 2. PSO algorithm with constriction factor

```
Input:
  $S$: size of the swarm,
  $D$: dimensions of the search space,
  $l,\ h$: lower and upper bound of the search space.
Method:
1: Initialize 1000 particles with random positions and velocities in the search space $[l, h]^D$, and then choose the best $S$
   particles to form the initial swarm;
2: repeat
3:   Update the swarm's fitness, $pbest_i$ and $gbest$ using Algorithm 1;
4:   Update the velocities and positions of the particles according to the Eqs. (2)–(4);
5: until A termination criterion is met (usually a sufficiently good fitness or a maximum number of iterations)
6: return The global best position $gbest$ and its fitness value.
```

3. The effects of the randomness in PSO and four PSO variants

It is obvious that the PSO algorithm is a stochastic algorithm as it has two random coefficients in the velocity update formulas, which are shown in Eqs. (1) and (3). The randomness is believed to have great importance to the PSO, but few researchers had studied its effects thoroughly. Besides, most of the improvements to the PSO preserve the randomness. In this section, the effects of the randomness and why it is indispensable for the traditional PSO are analyzed. Also, the methods to remove the random coefficients are discussed.

3.1. The importance of the randomness in the PSO

From the velocity updating formula, it can be seen that the particles will fly towards their personal best positions and the global best position of the whole swarm. But they do not fly straightly to the best solution, as there are

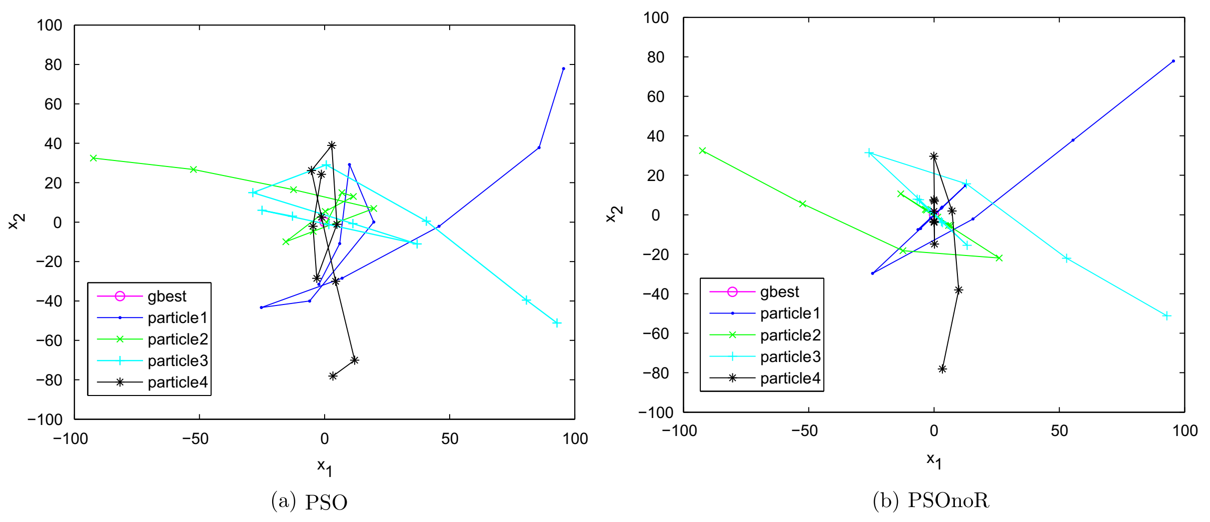

Fig. 1. Particles' movements in PSO and PSOnoR algorithms. The function to be minimized is $f_1(x)$ in Eq. (9), and the dimension of the search space is 2.

stochastic factors that influence their ability to learn from the social optimum and their own experience. The particles' movements in the PSO algorithm can be seen in Fig. 1(a). It is clear that in the PSO algorithm, the particles movements are more complex than that in PSOnoR algorithm (Fig. 1(a)) and can explore more area in the search space, which proved the importance of the random coefficients. The result of the randomness is that the particle will search a large area around the current best solution and probably find a better solution. The randomness guarantees a good balance between straying off-course and staying close to the optimal solution. This type of behavior seems to be ideal when exploring large error surfaces. Some particles can explore far beyond the current optimum, while the population still remembers the global best solution. This can also keep a population with certain degree of diversity, preventing the swarm from becoming premature.

In the process of finding a better solution, the PSO displays a "*two steps forward, one step back*" phenomenon [25,26], and some researchers [25–27] try to reduce the effects of this phenomenon by cooperative learning among the particles in the swarm. A similar example of this phenomenon as in [25,26] is described as follows. Given a problem, $f_0(x)$, which is defined in Eq. (5), and it is to be maximized.

$$f_0(x) = \sum_{i=1}^{D} x_i. \tag{5}$$

Let the dimension of the search space be 3, the search range as $[0, 20]^3$. Obviously, the optimal solution is $(20, 20, 20)$. Consider a swarm containing two particles, X_i and X_j, and X_i's three components x_{i1}, x_{i2} and x_{i3} is 18, 5 and 18 respectively, three components of X_j is 4, 20 and 4 respectively. So, X_i is the current global best solution, and X_j will fly towards X_i. In the next iteration, X_j will be updated and suppose its new value is $(16, 10, 16)$. The new value of X_j is better than its previous value, this will be considered as an improvement. However, the valuable information contained in the middle component of X_j has been lost, as x_{j2} previously is the same value as required in the optimal solution $(20, 20, 20)$. The reason for this behavior is that the error function is computed only after all the components in the solution vector have been updated to their new values. This means an improvement in two components (two steps forward) will overrule a potentially good value for a single component (one step back).

The PSO has certain ability to overcome the "*two steps forward, one step back*" problem through the random factor r_{1d} and r_{2d} in its velocity update formulas (Eqs. (1) and (3)). Because r_{1d} and r_{2d} can be any values between 0 and 1 with certain probability, so when a particle updates its position in one iteration, there are many different possibilities. Some cases are given below:

- In case 1, suppose $r_2 = (r_{21}, r_{22}, r_{23}) = (1, 0, 1)$, which means that X_j only learns from the best solution X_i's first and third dimensions, and does not learn from X_i's second dimension, and the new position for X_j maybe $(16, 20, 16)$.
- In case 2, suppose $r_2 = (0, 1, 0)$, which means that X_j does not learn from the best solution's first and third dimension, and only learn from the X_i's second dimension, and the new position for X_j maybe $(4, 10, 4)$.
- In case 3, suppose $r_2 = (1, 1, 1)$, which means that X_j learns from all of the best solution X_i's three dimensions, as is discussed in the "*two steps forward, one step back*" phenomenon, and the resulting new position for X_j maybe $(16, 10, 16)$.

Of course, there are other possibilities. In the three cases described above, the case 1 can get the best solution, and it is the solution that has the best fitness value can become the global best solution and provide search information for the whole swarm. So the case 1 is the least affected by the "*two steps forward, one step back*" problem and can be selected by the PSO as the new best solution. So the PSO can reduce the effects of the "*two steps forward, one step back*" problems to some extent, although not completely. The effect of the random coefficient r_1 is similar. This is a reason why randomness is important for the PSO.

3.2. Directly eliminate the randomness of the canonical PSO

To demonstrate the importance of the randomness in the PSO, a new form of PSO, PSO algorithm with no randomness (PSOnoR) is given. When a particle in the swarm updates its velocity on every dimension, there is no stochastic coefficient r_{1d} and r_{2d}, and they are replaced by their expectations as are used in many theoretical models for the PSO. r_{1d}^t and r_{2d}^t are random real numbers which are uniformly distributed on $[0,1]$, so both their expectations are 0.5. Then we can get the new form of the velocity update formula as shown in Eq. (6).

$$v_{id}^{t+1} = \chi \times \left(v_{id}^t + c_1 \times 0.5 \times \left(pbest_{id}^t - x_{id}^t\right) + c_2 \times 0.5 \times \left(gbest_d^t - x_{id}^t\right)\right). \tag{6}$$

In fact, PSOnoR cannot get acceptable results with any constant coefficients, we have tested the values $[0.01, 0.05, 0.1, 0.2, ., 0.9, 1]$, none of these values can get acceptable results. The process of the PSOnoR algorithm is the same as the PSO in Algorithm 2, except it uses the new velocity update formula which is shown in Eq. (6). From the discussion in Section 3.1, it can be seen that the randomness has essential effects for the PSO, and we speculate that the PSOnoR algorithm may not perform well on the optimization problems. The experiment results will be shown later.

3.3. PSO algorithm with random dimension selection

As discussed in Section 3.1, the randomness guarantees the swarm flies towards the best solutions while search a larger area in the search space around the best solution. Also, the stochastic coefficient can ensures the swarm keeps a high degree of diversity and prevents the swarm converge to the local optimum. Most importantly, the randomness can reduce the harm of the "*two steps forward, one step back*" problem. Simply eliminate the randomness without taking any measures to make up for that as in the PSOnoR algorithm may result in a worse performance. To maintain the advantage of the randomness in the traditional PSO, a new method is proposed. Instead of using stochastic coefficients in velocity update process on every dimension, some dimensions are randomly selected for every particle in every iteration. These selected dimensions will update their velocities according to a new formula without random coefficients, which is shown in Eq. (7). Then these dimensions will update its positions according to Eq. (2). Every particle's every dimension is selected at a certain probability. The rest dimensions, which are not selected in this iteration, will keep their old positions and velocities without any modification.

$$v_{id}^{t+1} = \chi \times \left(v_{id}^t + c_1 \times \left(pbest_{id}^t - x_{id}^t\right) + c_2 \times \left(gbest_d^t - x_{id}^t\right)\right). \tag{7}$$

The specific algorithm of PSO with random dimension selection (PSORDS) is shown in Algorithm 3.

Algorithm 3. PSO algorithm with random dimension selection (PSORDS)

```
Input:
  S: size of the swarm,
  D: dimensions of the search space,
  l, h: lower and upper bound of the search space.
Method:
1: Initialize 1000 particles with random positions and velocities in the search space $[l,h]^D$, and then choose the best S
   particles to form the initial swarm;
2: repeat
3:   Update the swarm's fitness, $pbest_i$ and gbest using Algorithm 1;
4:   Every particle's every dimension is selected at a certain probability;
5:   For every particle's every selected dimension, update its velocity and position according to Eqs. (2), (4) and (7). For
   the dimensions that are not selected, their velocities and positions will keep unchanged;
6: until A termination criterion is met (usually a sufficiently good fitness or a maximum number of iterations)
7: return The global best position gbest and its fitness value.
```

3.4. PSO algorithm with heuristic dimension selection

As the randomness has significant impacts on the performance of the PSO, simply eliminating it would cause a deteriorated performance. So the PSORDS algorithm uses the random dimension selection strategy to reduce the effects of removing the stochastic coefficients. But in the process of the dimension selection, dimensions are selected randomly, so the PSORDS algorithm does not eliminate the randomness completely. To further reduce the randomness, a heuristic strategy for selecting the dimensions is given. In the heuristic strategy, the current global worst particle *gworst* and the global best solution *gbest* are used, specific details are given in Rule 1.

Rule 1. Denote the global best position as *gbest* and global worst position as *gworst*. Let D be the dimensions of the search space, for each dimension d in D, replace the $gworst_d$ with $gbest_d$, if the new value $gworst'$ has a better fitness, that is $f(gworst') < f(gworst)$, then dimension d is selected.

The dimension selection process is taken when the *gbest* has changed. In this strategy it is supposed that a dimension of the *gbest*'s position vector is a real optimal value if we can get a better solution by replace the *gworst*'s position's corresponding dimension with it. If the supposition is right, then the "*two steps forward, one step back*" problem can be eliminated, because only the best dimensions of the *gbest* are selected. There is only "*two steps forward*" and no "*one step back*". The specific algorithm of the PSO with heuristic dimension selection (PSOHDS) is shown in Algorithm 4.

Algorithm 4. PSO algorithm with heuristic dimension selection (PSOHDS)

```
Input:
  S: size of the swarm,
  D: dimensions of the search space,
  l, h: lower and upper bound of the search space.
Method:
1: Initialize 1000 particles with random positions and velocities in the search space $[l,h]^D$, and then choose the best S
   particles to form the initial swarm;
2: repeat
3:    Update the swarm's fitness, $pbest_i$ and gbest using Algorithm 1;
4:    If the gbest has changed, then execute the dimension selection process according to Rule 1;
5:    If a dimension is selected, for every particle, update this dimension's velocity and position according to Eqs. (2), (4)
   and (7). For the dimensions that are not selected, their velocities and positions will keep unchanged;
6: until A termination criterion is met (usually a sufficiently good fitness or a maximum number of fitness evaluations)
7: return The global best position gbest and its fitness value.
```

3.5. Dimension selection based on difference from the global best individual

In Section 3.4, dimension selection is based on the global best individual and global worst individual, and a dimension is selected if it can help the global worst particle to improve its position. Although this selection method takes advantage of more information than the random dimension selection method, the dimensions selected maybe not suitable for all individuals in the swarm. So, instead of selecting dimensions for the whole swarm, it could be more appropriate that every particle in the swarm selects its own dimensions that need to learn from the *gbest* based on its own characteristics. Thus, a method that select the dimensions for each particle base on their difference from the *gbest* is proposed. The dimension selection is done according to Rule 2.

Rule 2. For each particle x_i, compute its mean distance from the *gbest*, as shown in Eq. (8).

$$mdistance_i = \frac{1}{D}\sum_{d=1}^{D}\mathrm{abs}(gbest_d - x_{id}), \tag{8}$$

where D is the dimension of the search space. For each dimension d of particle x_i, if its distance to the *gbest*'s corresponding dimension is longer than mean distance $mdistance_i$, that is, $\mathrm{abs}(gbest_d - x_{id}) > mdistance_i$, then the dimension d is selected for particle x_i.

For every particle's every dimension, we use the Rule 2 to decide whether it is selected. The selected dimensions will update their velocities according to a formula without random coefficients, which is shown in Eq. (7). Then these dimensions will update its positions according to Eq. (2). The rest dimensions, which are not selected in this iteration, will keep their old positions and velocities without any modification. In this method, it is thought that if a particle's dimension is far away from the *gbest*, then learning from *gbest* is very urgent than the dimensions that are similar to the *gbest*. Also, as the more similar dimensions will not change their velocities and positions in this generation, it will help to keep the diversity of the swarm, preventing the premature convergence. PSO with distance based dimension selection algorithm is referred as PSODDS for short, its specific procedure is described in Algorithm 5.

Algorithm 5. PSO with distance based dimension selection (PSODDS)

```
Input:
  $S$: size of the swarm,
  $D$: dimensions of the search space,
  $l, h$: lower and upper bound of the search space.
Method:
1: Initialize 1000 particles with random positions and velocities in the search space $[l,h]^D$, and then choose the best $S$
   particles to form the initial swarm;
2: repeat
3:   Update the swarm's fitness, $pbest_i$ and $gbest$ using Algorithm 1;
4:   Using Rule 2 to choose the dimensions that need to update for each particle;
5:   For every particle's every selected dimension, update its velocity and position according to Eqs. (2), (4) and (7). For
     the dimensions that are not selected, their velocities and positions will keep unchanged;
6: until A termination criterion is met (usually a sufficiently good fitness or a maximum number of fitness evaluations)
7: return The global best position gbest and its fitness value.
```

4. Experiments and results

We have conducted intensive experiments to verify our analysis of the effects of the randomness in the PSO and to evaluate the performance of the proposed algorithms. Also, the dimension selection method is used to a newly proposed PSO variant and comparison of the results are given.

4.1. Test functions and experimental settings

Ten mathematical functions are chosen to test the proposed algorithms. These 10 test functions are given below, $f_1(x) \sim f_5(x)$ are 5 unimodal functions and $f_6(x) \sim f_{10}(x)$ are 5 complex multimodal functions with many local optima. All these functions are to be minimized. For PSOnoR, PSORDS and PSOHDS algorithms, the dimensions of the functions are all set to 30 and every function's search range, their actual minimum and the acceptable worst solutions are shown in Table 1. Following the settings in [28], for PSO, PSODDS, modified particle swarm optimization (MPSO) and MPSO with distance based dimension selection (MPSODDS) algorithms, for each benchmark function, three different dimension sizes are tested, which are 10, 20, 30.

$$f_1(x) = \sum_{i=1}^{D} x_i^2, \tag{9}$$

$$f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \tag{10}$$

$$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2, \tag{11}$$

Table 1
Specification of the 10 test functions.

Function	D	Search range	f_{min}	Accept
$f_1(x)$	30	$[-100, 100]^D$	0	0.01
$f_2(x)$	30	$[-10, 10]^D$	0	0.01
$f_3(x)$	30	$[-100, 100]^D$	0	200
$f_4(x)$	30	$[-100, 100]^D$	0	0.01
$f_5(x)$	30	$[-10, 10]^D$	0	100
$f_6(x)$	30	$[-500, 500]^D$	−12596.5	−5000
$f_7(x)$	30	$[-5.12, 5.12]^D$	0	150
$f_8(x)$	30	$[-32, 32]^D$	0	5
$f_9(x)$	30	$[-600, 600]^D$	0	1
$f_{10}(x)$	30	$[-50, 50]^D$	0	1

$$f_4(x) = \max\{|x_i|,\, 1 \leqslant i \leqslant D\}, \tag{12}$$

$$f_5(x) = \sum_{i=1}^{D-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right], \tag{13}$$

$$f_6(x) = \sum_{i=1}^{D} - x_i \sin\left(\sqrt{|x_i|}\right), \tag{14}$$

$$f_7(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right], \tag{15}$$

$$f_8(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e, \tag{16}$$

$$f_9(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D}\cos\left(x_i/\sqrt{i}\right) + 1, \tag{17}$$

$$f_{10}(x) = \pi/D\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4), \tag{18}$$

where,

$$y_i = 1 + (x_i - 1)/4,$$

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

The PSO related parameters for the traditional PSO and the proposed PSO variants are the same. That is, the c_1 and c_2 are both set to 2.05, and the constriction factor, χ, is approximately 0.7298. Moreover, every particle's velocities are constricted within 20% of the search range of the corresponding dimension. In the PSORDS algorithm, every dimension is selected with the probability of 0.5, as in the traditional PSO when a particle learns from the global best solution the random coefficient c_2's expectation is 0.5. To compare the performances of different methods, the PSO and its variants use the same population size of 40. Moreover, these algorithms use the same maximal number of 2×10^5 fitness evaluations (FEs) for each test function. For the PSO, PSOnoR, PSORDS and PSODDS algorithms, in every iteration the fitness evaluation times are the same, that is 40, but for the PSOHDS algorithms, when execute the dimension selection process, extra times of FE which is equal to the number of the dimensions is required, and this must be considered.

On the purpose of avoiding stochastic error, we simulate 25 independent trials on each test function and use the average, median, best and worst results and the standard deviation over 25 runs for comparisons. The success rate is also considered, which means the percentage of the runs that get acceptable solutions. Wilcoxon rank sum test [29] is used to show whether the difference between different algorithms' results is statistically significant. We perform a two-sided rank sum test of the null hypothesis that results of two algorithms are independent samples from identical continuous distributions with equal medians, against the alternative that they do not have equal medians. If a rejection of the null hypothesis at the 5% significance level is got, it is considered that the difference is statistically significant (using Y to denote statistically significant in Table 4).

Table 2
Experiment results for the PSO algorithm on the ten test functions.

Function	Success (%)	Best	Mean	Median	Worst	STD
$f_1(x)$	100	6.35E−107	9.06E−100	7.70E−103	1.30E−98	2.70E−99
$f_2(x)$	100	6.02E−52	1.35E−40	2.21E−45	2.04E−39	4.68E−40
$f_3(x)$	100	4.85E−13	2.53E−11	1.60E−11	1.05E−10	2.95E−11
$f_4(x)$	100	3.20E−08	1.01E−06	4.40E−07	7.79E−06	1.58E−06
$f_5(x)$	100	0.0184258	18.480248	11.668977	73.887093	23.396476
$f_6(x)$	100	−9547.685	−8108.587	−8068.612	−7180.167	615.84703
$f_7(x)$	100	25.86892	52.218198	48.752933	96.581798	16.656965
$f_8(x)$	100	7.99E−15	0.9541351	1.1551485	2.3161618	0.8572157
$f_9(x)$	100	0	0.0256187	0.0221561	0.1050763	0.0251739
$f_{10}(x)$	96	1.57E−32	0.1580123	1.60E-32	1.6656984	0.3717751

Table 3
Experiment results for the PSOnoR algorithm on the ten test functions.

Function	Success (%)	Best	Mean	Median	Worst	STD
$f_1(x)$	0	438.59	1013.68	913.23	1955.94	440.14
$f_2(x)$	0	11.06	17.76	17.06	29.34	4.02
$f_3(x)$	0	1184.72	4415.31	3585.51	11837.45	2822.19
$f_4(x)$	0	12.3	18.84	19.14	29.22	4.37
$f_5(x)$	0	302.82	2493.24	2054.58	7520.08	1798
$f_6(x)$	60	−7502.67	−5462.19	−5309.72	−3792.24	969.87
$f_7(x)$	92	57.57	102.89	101.38	163.56	29.33
$f_8(x)$	0	6.41	9.53	9.6	11.72	1.35
$f_9(x)$	0	3.37	11.2	9.88	28.77	5.65
$f_{10}(x)$	0	6.97	34.17	25.39	163.4	38.06

Table 4
Performance comparison of different algorithms. For each algorithm, the mean result are given. Also, Wilcoxon rank sum test is used to show whether the result difference between the PSO and PSO variants is statistically significant at the 5% significance level, Y denotes significant, N denotes not. (Y does not implies that one algorithm is better than the other, just means that the two are significantly different.)

Function	PSO	PSOnoR		PSORDS		PSOHDS		PSODDS	
$f_1(x)$	9.06E−100	*1013.68*	Y	*9.08E−35*	Y	6.88E−102	N	*1.36E−81*	Y
$f_2(x)$	1.35E−40	*17.76*	Y	*2.38E−18*	Y	**6.79E−54**	Y	**2.31E−43**	Y
$f_3(x)$	2.53E−11	*4415.31*	Y	*1.12E−06*	Y	*74.919185*	Y	**2.11E−21**	Y
$f_4(x)$	1.01E−06	*18.84*	Y	*7.97E−05*	Y	*76.828155*	Y	**7.60E−09**	Y
$f_5(x)$	18.480248	*2493.24*	Y	*28.044785*	Y	33.469738	N	**1.1162856**	Y
$f_6(x)$	−8108.587	*−5462.19*	Y	*−7328.097*	Y	*−6506.112*	Y	−7984.568	N
$f_7(x)$	52.218198	*102.89*	Y	61.093211	N	*80.28489*	Y	58.264668	N
$f_8(x)$	0.9541351	*9.53*	Y	**0.0924119**	Y	1.047658	N	**0.1062758**	Y
$f_9(x)$	0.0256187	*11.2*	Y	**0.0131507**	Y	*0.3773746*	Y	0.0144671	N
$f_{10}(x)$	0.1580123	*34.17*	Y	0.0124403	N	**0.1021139**	Y	**0.1368918**	Y

4.2. Results of the PSO and PSOnoR algorithms

The results of the PSO and PSOnoR algorithms on the ten test functions are shown in Tables 2 and 3, where success means the success rate and STD means the standard deviation. Almost every run of the PSO algorithm on every function can get an acceptable result. However, the PSOnoR algorithm can hardly get an acceptable result, which is the same as speculated. Also, as shown in Table 4, the Wilcoxon rank sum test demonstrates that PSOnoR is significantly poorer than PSO. Without randomness, the traditional PSO cannot find a right solution for most of the problems. The PSOnoR algorithm eliminates the random coefficients, which means that every particle flies toward the global best position and its personal best solution on every dimension, regardless of whether the dimension of the global best has a real better value. So the swarm will lose its diversity rapidly. In fact, the particles' velocities in the PSOnoR algorithm reduce to less than 0.001 after 200 iterations, which is a premature convergence. Also, the PSOnoR algorithm cannot take advantage of the randomness and reduce the harm effects of the "*two steps forward, one step back*" problem as discussed in Section 3.1, which also decreased its ability to find a better solution. The poor performance of the PSOnoR algorithm conforms to the analysis of the importance about randomness in Section 3.1.

Table 5
Experiment results for the PSORDS algorithm on the ten test functions.

Function	Success (%)	Best	Mean	Median	Worst	STD
$f_1(x)$	100	4.14E−38	9.08E−35	1.26E−35	1.11E−33	2.28E−34
$f_2(x)$	100	4.71E−20	2.38E−18	6.46E−19	2.26E−17	4.93E−18
$f_3(x)$	100	3.82E−09	1.12E−06	2.44E−07	1.50E−05	2.99E−06
$f_4(x)$	100	4.29E−06	7.97E−05	3.20E−05	0.0003956	0.000108
$f_5(x)$	96	0.0727298	28.044785	15.495931	315.79205	61.139181
$f_6(x)$	92	−9608.363	−7328.097	−7512.655	−3873.635	1331.6239
$f_7(x)$	100	32.833629	61.093211	57.70753	127.35427	20.087955
$f_8(x)$	100	1.15E−14	0.0924119	2.22E−14	1.1551485	0.3198461
$f_9(x)$	100	0	0.0131507	0.0098573	0.0633896	0.0179166
$f_{10}(x)$	100	1.57E−32	0.0124403	1.61E−32	0.103669	0.0343831

4.3. Results of the PSORDS algorithm

The results of the PSORDS algorithm are shown in Table 5. Because the PSO is a stochastic algorithm, some bad results can substantially influence the average and standard deviation of the results, so the success rate, the best solution and the median solution can better indicate the algorithm's performance. Most of the median results are better than the average results. Also, as shown in Table 4, the Wilcoxon rank sum test gives a more clear comparison of PSO and PSORDS. On the unimodal functions $f_1(x) \sim f_5(x)$, PSO algorithm gets better results. But it must be noted that PSORDS also gets excellent results, for $f_1(x) \sim f_4(x)$, the errors from true optimal value is very small, which are less than 1×10^{-5}. For the multimodal functions, PSO and PSORDS get similar results for functions $f_7(x)$ and $f_{10}(x)$, and PSO is better than PSORDS for $f_6(x)$ while PSORDS is better than PSO for $f_8(x)$ and $f_9(x)$. This proves the PSORDS algorithm has considerable capabilities to find a good solution. Although without stochastic coefficients, in PSORDS algorithm the swarm can keep a high degree of diversity by the random selection of the dimensions to be updated. This guarantees the swarm can search a large area for potential solutions while flying towards the current best solution. Through the random dimension selection, the algorithm can also reduce the detriment of the "*two steps forward, one step back*" problem.

4.4. Results of the PSOHDS algorithm

The results of the PSOHDS algorithm are displayed in Table 6. Also, Wilcoxon rank sum test gives a more clear comparison of PSO and PSOHDS, which is shown in Table 4. PSO and PSOHDS algorithms get similar results for functions $f_1(x),\ f_5(x)$ and $f_8(x)$, PSOHDS gets better results for functions $f_2(x)$ and $f_{10}(x)$, and PSO algorithm gets better results on other functions. PSOHDS can get comparable results for most of the test functions, while for functions $f_3(x)$ and $f_4(x)$, the results are much worse. The well performance of PSOHDS algorithm demonstrates that it is possible to eliminate the random coefficients of the PSO algorithm. Through the heuristic dimension selection strategy, the particles can avoid converge to the bad dimensions of the current best solution, which can prevent the fast losing of diversity. Also, the particles only update its positions according to the good dimensions of the current best solution, and this can guarantee there are only "two steps forward" and no "one step back", which increase the probability of finding a better solution. But the results of PSOHDS algorithm show no superiority to PSO algorithm, this possibly because the heuristic dimension strategy can not ensure that all the good dimensions are selected and none of the bad dimensions is selected. In fact, the used strategy perceives the dimensions that cause better results for the global worst particle as the good dimensions for the whole swarm, but this is possibly not the case. Function $f_4(x)$ is the evidence that the strategy is not a perfect one, as the PSOHDS algorithm cannot find a correct solution. So better strategies for dimension selection need to be proposed.

Table 6
Experiment results for the PSOHDS algorithm on the ten test functions.

Function	Success (%)	Best	Mean	Median	Worst	STD
$f_1(x)$	100	6.07E−106	6.88E−102	1.84E−103	4.60E−101	1.24E−101
$f_2(x)$	100	4.03E−56	6.79E−54	1.76E−54	4.82E−53	1.10E−53
$f_3(x)$	96	13.818195	74.919185	57.54334	229.14366	56.067009
$f_4(x)$	0	68.916951	76.828155	77.255096	81.751701	3.1926958
$f_5(x)$	96	0.1086075	33.469738	16.247792	152.88202	39.6307
$f_6(x)$	88	−8582.52	−6506.112	−6511.663	−3886.202	1092.4477
$f_7(x)$	100	40.26331	80.28489	80.012511	133.666	25.112593
$f_8(x)$	100	4.35E−14	1.047658	1.1551485	2.2201128	0.7344387
$f_9(x)$	100	0.00E+00	0.3773746	0.4207715	0.9167278	0.2910044
$f_{10}(x)$	96	1.59E−32	0.1021139	1.84E−32	1.4339354	0.2878731

Table 7
Experiment results for the PSODDS algorithm on the ten test functions.

Function	Success (%)	Best	Mean	Median	Worst	STD
$f_1(x)$	100	4.04E−84	1.36E−81	4.69E−82	1.13E−80	2.77E−81
$f_2(x)$	100	2.13E−44	2.31E−43	1.01E−43	1.32E−42	3.36E−43
$f_3(x)$	100	3.78E−24	2.11E−21	3.71E−22	2.23E−20	4.71E−21
$f_4(x)$	100	1.79E−11	7.60E−09	6.78E−10	8.69E−08	2.04E−08
$f_5(x)$	100	4.07E−08	1.1162856	4.24E−05	3.9866722	1.8268891
$f_6(x)$	100	−9052.755	−7984.568	−8007.99	−6854.148	607.01625
$f_7(x)$	100	36.81344	58.264668	57.70755	78.601548	10.697031
$f_8(x)$	100	1.51E−14	0.1062758	2.22E−14	1.5017466	0.3712169
$f_9(x)$	100	0	0.0144671	0.012321	0.0541378	0.01358
$f_{10}(x)$	100	1.65E−32	0.1368918	2.56E−32	0.8299968	0.2294781

Table 8
Experiment results for PSO and PSODDS algorithms for the ten functions with different dimensions. For each algorithm, the mean result are given. Also, Wilcoxon rank sum test is used to show whether the result difference between PSO and PSODDS algorithm is statistically significant at the 5% significance level, Y denotes significant, N denotes not. (Y does not implies that one algorithm is better than the other, just means that the two are significantly different.)

D	10 dimensions			20 dimensions			30 dimensions		
	PSO	PSODDS		PSO	PSODDS		PSO	PSODDS	
$f_1(x)$	**1.63E−152**	8.43E−81	Y	**3.86E−154**	5.48E−109	Y	**9.06E−100**	1.36E−81	Y
$f_2(x)$	**2.45E−81**	6.28E−41	Y	**6.70E−76**	4.45E−55	Y	1.35E−40	**2.31E−43**	Y
$f_3(x)$	3.97E−71	**1.68E−71**	Y	1.74E−30	**5.03E−48**	Y	2.53E−11	**2.11E−21**	Y
$f_4(x)$	**2.20E−54**	1.89E−39	Y	1.93E−23	**2.96E−28**	Y	1.01E−06	**7.60E−09**	Y
$f_5(x)$	0.3436055	**1.59E−01**	Y	1.8498873	**0.31893**	Y	18.48	**1.11629**	Y
$f_6(x)$	**−3480.7**	−3079.282	Y	−5943.327	−5722.115	N	−8108.5	−7984.568	N
$f_7(x)$	7.2034981	7.8402711	N	28.415991	33.510168	N	52.22	58.26	N
$f_8(x)$	**4.44E−15**	5.58E−15	Y	0.2065569	**1.30E−14**	Y	0.954	**0.106**	Y
$f_9(x)$	**0.07015**	0.1386538	Y	0.0379026	**0.023063**	Y	0.0256	0.014	N
$f_{10}(x)$	4.71E−32	**4.54E−32**	Y	0.0435384	**0.01866**	Y	0.158	**0.137**	Y

Table 9
Experiment results for MPSO and MPSODDS algorithms for the ten functions with different dimensions. For each algorithm, the mean result are given. Also, Wilcoxon rank sum test is used to show whether the result difference between MPSO and MPSODDS algorithm is statistically significant at the 5% significance level, Y denotes significant, N denotes not. (Y does not implies that one algorithm is better than the other, just means that the two are significantly different.)

D	10 dimensions			20 dimensions			30 dimensions		
	MPSO	MPSODDS		MPSO	MPSODDS		MPSO	MPSODDS	
$f_1(x)$	**1.82E−186**	1.16E−63	Y	**1.3E−194**	1.398E−93	Y	**2.6E−110**	1.579E−69	Y
$f_2(x)$	**1.34E−60**	1.93E−43	Y	3.239E−08	**4.34E−51**	Y	1.042E−06	**1.96E−42**	Y
$f_3(x)$	**1.32E−80**	1.03E−66	Y	8.196E−12	**4.83E−42**	Y	0.0894217	**6.62E−18**	Y
$f_4(x)$	**6.80E−81**	4.39E−40	Y	5.41E−22	**3.63E−22**	Y	8.559E−06	**8.39E−09**	Y
$f_5(x)$	0.0007567	**2.38E−05**	Y	0.0088749	**1.16E−06**	Y	0.8755834	**0.604493**	Y
$f_6(x)$	−1230.404	**−3412.65**	Y	46.882519	**−6195.67**	Y	44.918094	**−8500.64**	Y
$f_7(x)$	**5.894367**	23.539226	Y	**13.95048**	104.49795	Y	**51.98043**	187.44934	Y
$f_8(x)$	6.72E−15	**5.00E−15**	Y	1.965E−14	1.368E−14	N	5.304E−14	**2.23E−14**	Y
$f_9(x)$	**0.021262**	0.2014979	Y	0.048	0.0516112	N	0.0198142	0.0275372	N
$f_{10}(x)$	4.71E−32	4.90E−32	N	**2.36E−32**	2.623E−32	Y	**1.57E−32**	0.0540318	Y

4.5. Results of the PSODDS algorithm

The results of the PSODDS algorithm are displayed in Table 7. Following the settings in [28], for each benchmark function, three different dimension sizes are tested, which are 10, 20, 30, and the corresponding maximum number of generations are set as 3000, 5000 and 5000, other parameters are the same as described in Section 4.1. Also, Wilcoxon rank sum test gives a more clear comparison of PSO and PSODDS (the test setting is given in Section 4.1), the results are given in Table 8 (also in Table 4 for the 30 dimensions case). For the 30 dimensions case, it can be seen that PSODDS is considerably superior to PSO algorithm. Specifically, PSODDS gets better results for functions $f_2(x) \sim f_5(x)$, $f_8(x)$ and $f_{10}(x)$, and similar results to PSO are got for functions $f_6(x)$, $f_7(x)$ and $f_9(x)$. PSO only gets better results than PSODDS for $f_1(x)$, however, PSODDS's result is also very satisfying, its error is less than 1×10^{-81}. PSODDS is also much better than PSO for the 20 dimensions case. However, for the simple 10 dimensions functions, the PSO is better than PSODDS. The perfect performance of PSODDS algorithm for complex functions demonstrates that it is possible to eliminate the random coefficients of the PSO algorithm. Through the distance based dimension selection strategy, the particles only learn from these dimensions that can give great help for the particles, while other dimensions keep unchanged, which can keep the diversity of the swarm and prevent the premature convergence.

4.6. Using dimension selection methods in recently proposed PSO variant

To show the usefulness of the dimension selection methods proposed in Section 3.5, we apply it to a newly proposed PSO variant. In [30], a modified particle swarm optimization algorithm (MPSO) is developed, in which the mean value of past optimal positions (in Eq. (19)) for each particle and the mutation operation (Rule 3) are considered for avoiding premature. The specific procedure of MPSO algorithm can be found in [30]. Apply the dimension selection method into MPSO, the resultant MPSODDS algorithm can be found, its specific procedure is in Algorithm 6.

$$mbest = \frac{1}{n}\sum_{i=1}^{n} pbest_i, \tag{19}$$

Rule 3. The mutation take place with mutation probability P_m, each time select D_m dimension to mutate. Let $xmin_d = \min_i\{x_{id}\},\ xmax_d = \max_i\{x_{id}\}$, if a dimension d is selected to mutate, than $gbest_d = xmin_d + r \times (xmax_d - xmin_d)$, where r is uniformly distributed random number in the interval $(0, 1)$.

Algorithm 6. MPSO with distance based dimension selection (MPSODDS)

```
Input:
  $S$: size of the swarm,
  $D$: dimensions of the search space,
  $l,\ h$: lower and upper bound of the search space,
  $P_m,\ D_m$: mutation probability and the number of dimensions need to mutate each time.
Method:
1: Initialize 1000 particles with random positions and velocities in the search space $[l, h]^D$, and then choose the best $S$
   particles to form the initial swarm;
2: repeat
3:    Update the swarm's fitness, $pbest_i$ and $gbest$ using Algorithm 1;
4:    Renew $pbest_i$ and $gbest$ according to Eq. (19) and Rule 3 respectively.
5:    Using Rule 2 to choose the dimensions that need to update for each particle;
6:    For every particle's every selected dimension, update its velocity and position according to Eqs. (2), (4) and (7). For
   the dimensions that are not selected, their velocities and positions will keep unchanged;
7: until A termination criterion is met (usually a sufficiently good fitness or a maximum number of fitness evaluations)
8: return The global best position $gbest$ and its fitness value.
```

Following the settings in [28], for each benchmark function, three different dimension sizes are tested, which are 10, 20, 30, and the corresponding maximum number of generations are set as 3000, 5000 and 5000, the population size is 40. Same as in MPSO [30], the inertia weight is linearly decreased from 0.9 to 0.4, $c_1 = c_2 = 1.4945$. The mutation probability P_m is linearly decreased from 0.1 to 0.01. The number of dimensions that need to mutate every time is D_m = 5, 5, 2 for 30, 20, 10 dimension test functions respectively. Also, Wilcoxon rank sum test gives a more clear comparison of MPSO and MPSODDS (the test setting is given in Section 4.1), the results are given in Table 9. It can be seen that the MPSODDS is more superior than MPSO for the 20 and 30 dimension cases, while MPSO got better results for the much simpler 10 dimension case. So, the dimension selection method is very useful.

5. Conclusions

The randomness in the PSO algorithm plays an essential role in guaranteeing the PSO's performance, and it presents considerable obstacle for the researchers who want to build a theoretical model for the PSO. This paper analyzes the importance of the randomness in the PSO and explains why randomness is important. Then PSOnoR algorithm, a PSO variant without randomness, is presented to demonstrate that the randomness is indispensable for the traditional PSO. Based on our analysis, an equivalent stochastic PSO algorithm, PSORDS, is presented to show another way of using the randomness. Dimension selection using more deterministic methods, i.e., dimension selection based on global best and worst particles and based on distance to the global best particle, resulting in PSOHDS and PSODDS algorithms, are discussed. Experiments show that PSOHDS is comparable with traditional PSO on most of the test functions and PSODDS is greatly superior to traditional PSO algorithm. In addition, using our dimension selection method in a newly proposed MPSO algorithm also gets improved results. Experiment results proved the analysis about the randomness in the paper is correct and the effort to eliminate the random coefficients and use more deterministic dimension selection methods in the PSO is helpful.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61175052, 60975039, 61203297, 60933004, 61035003, 71240003), Natural Science Foundation of Shandong Province, China (No. ZR2012FM003), National High-tech R&D Program of China (863 Program) (No.2012AA011003) and National Program on Key Basic Research Project (973 Program) (No. 2013CB329502).

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, IEEE International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942–1948.
[2] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence 1 (1) (2007) 33–57.
[3] X. Xie, W. Zhang, Z. Yang, Overview of particle swarm optimization, Control and Decision 18 (2) (2003) 129–134.
[4] E. Ozcan, C. Mohan, Analysis of a simple particle swarm optimization system, Intelligent Engineering Systems through Artificial Neural Networks 8 (1998) 253–258.
[5] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.

[6] K. Yasuda, A. Ide, N. Iwasaki, Adaptive particle swarm optimization, IEEE International Conference on Systems, Man and Cybernetics, vol. 2, IEEE, 2003, pp. 1554–1559.
[7] B. Brandstatter, U. Baumgartner, Particle swarm optimization–mass-spring system analogon, IEEE Transactions on Magnetics 38 (2) (2002) 997–1000.
[8] J. Zeng, Z. Cui, A new unified model of particle swarm optimization and its theoretical analysis, Journal of Computer Research and Development 1 (2006) 96–100.
[9] W. Hu, Z. Li, Simpler and more effective particle swarm optimization algorithm, Ruan Jian Xue Bao (Journal of Software) 18 (4) (2007) 861–868.
[10] R. He, Y. Wang, Q. Wang, J. Zhou, C. Hu, Improved particle swarm optimization based on self-adaptive escape velocity, Ruan Jian Xue Bao (Journal of Software) 16 (12) (2005) 2036–2044.
[11] C. Kurosu, T. Saito, K. Jinno, Growing particle swarm optimizers with a population-dependent parameter, in: Neural Information Processing, Springer, 2009, pp. 234–241.
[12] C. Zhang, J. Sun, D. Ouyang, Y. Zhang, A self-adaptive hybrid particle swarm optimization algorithm for flow shop scheduling problem, Chinese Journal of Computers 11 (2009) 2137–2146.
[13] S. Yu, Z. Wu, H. Wang, Z. Chen, A hybrid particle swarm optimization algorithm based on space transformation search and a modified velocity model, High Performance Computing and Applications (2010) 522–527.
[14] P. Kim, J. Lee, An integrated method of particle swarm optimization and differential evolution, Journal of Mechanical Science and Technology 23 (2) (2009) 426–434.
[15] P. Yin, F. Glover, M. Laguna, J. Zhu, Cyber swarm algorithms-improving particle swarm optimization using adaptive memory strategies, European Journal of Operational Research 201 (2) (2010) 377–389.
[16] M. Chen, X. Li, X. Zhang, Y. Lu, A novel particle swarm optimizer hybridized with extremal optimization, Applied Soft Computing 10 (2) (2010) 367–373.
[17] H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, Applied Soft Computing 10 (2) (2010) 629–640.
[18] L. Cartwright, T. Hendtlass, A heterogeneous particle swarm, Artificial Life: Borrowing from Biology (2009) 201–210.
[19] C. Hu, B. Wang, Y. Wang, Multi-swarm particle swarm optimiser with cauchy mutation for dynamic optimisation problems, International Journal of Innovative Computing and Applications 2 (2) (2009) 123–132.
[20] Q. Ni, Z. Zhang, Z. Wang, H. Xing, Dynamic probabilistic particle swarm optimization based on varying multi-cluster structure, Ruan Jian Xue Bao (Journal of Software) 20 (2) (2009) 339–349.
[21] J. Jie, J. Zeng, C. Han, Self-organized particle swarm optimization based on feedback control of diversity, Jisuanji Yanjiu yu Fazhan (Computer Research and Development) 45 (3) (2008) 464–471.
[22] X. Zhao, A perturbed particle swarm algorithm for numerical optimization, Applied Soft Computing 10 (1) (2010) 119–124.
[23] Z. Zhan, J. Zhang, Y. Li, H. Chung, Adaptive particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39 (6) (2009) 1362–1381.
[24] T. Chen, T. Chi, On the improvements of the particle swarm optimization algorithm, Advances in Engineering Software 41 (2) (2010) 229–239.
[25] F. van den Bergh, A. Engelbrecht, Cooperative learning in neural networks using particle swarm optimizers, South African Computer Journal (2000) 84–90.
[26] F. Van den Bergh, A. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 225–239.
[27] Z. Zhan, J. Zhang, Parallel particle swarm optimization with adaptive asynchronous migration strategy, Algorithms and Architectures for Parallel Processing (2009) 490–501.
[28] G. Ma, W. Zhou, X. Chang, A novel particle swarm optimization algorithm based on particle migration, Applied Mathematics and Computation 218 (11) (2012) 6620–6626, http://dx.doi.org/10.1016/j.amc.2011.12.032.
[29] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.
[30] G. He, N. Huang, A modified particle swarm optimization algorithm with applications, Applied Mathematics and Computation 2012, http://dx.doi.org/10.1016/j.amc.2012.07.010, <http://www.sciencedirect.com/science/article/pii/S0096300312007102>.